Referring Expression Comprehension

Bachelor's Thesis

David Álvarez Rosa

This page intentionally left blank.

Exploring and Visualizing Referring Expression Comprehension

University of Toronto

Politechnical University of Catalonia

Supervisor
Sanja Fidler

Co-Supervisor
Xavier Giró

Exploring and Visualizing Referring Expression Comprehension

A senior Bachelor's Degree Thesis written by
David Álvarez Rosa

in partial fullfilment of the requirements for the bachelors' degrees in
MATHEMATICS
INDUSTRIAL TECHNOLOGY ENGINEERING

and submitted to
INTERDISCIPLINARY HIGHER EDUCATION CENTRE
BARCELONA SCHOOL OF INDUSTRIAL ENGINEERING
SCHOOL OF MATHEMATICS AND STATISTICS

Toronto, ON, Canada

May 25, 2021

Exploring and Visualizing Referring Expression Comprehension
`https://recomprehension.com`

Thesis typeset with pdfTeX 3.14159265–2.6–1.40.21 (TeX Live 2020) on Arch Linux using Latin Modern typefaces and written with GNU Emacs. The BibLaTeX package has been used for bibliography management with Biber as processing backend.

Vector graphics have been created by the author using PGF/Ti*k*Z. Vectorian decorative ornaments are from the LaTeX package `pgfornament`.

To my mother, for her love and patience.
To all my friends.

Exploring and Visualizing Referring Expression Comprehension

by David Álvarez Rosa

Abstract

Human-machine interaction is one of the main objectives currently in the field of Artificial Intelligence. This work will contribute to enhance this interaction by exploring the new task of Referring Expression Comprehension (REC), consisting of: given a referring expression—which can be a linguistic phrase or human speech—and an image, detect the object to which the expression refers (i.e., achieve a binary segmentation of the referred object). The multimodal nature of this task will require the use of different deep learning architectures, among them: convolutional neural networks (computer vision); and recurrent neural networks and the Transformer model (natural language processing).

This thesis is presented as a self-contained document that can be understood by a reader with no prior knowledge of machine learning. The bulk of the work consists of an exhaustive study of the REC task: from the applications; until the study, comparison and implementation of models; going through a complete description of the current state of the art. Likewise, a functional, free and public web page is presented in which interaction is allowed in a simple way with the model described in this work.

Keywords

Referring Expression Comprehension
Artificial Intelligence • Machine Learning • Deep Learning
Computer Vision • Natural Language Processing
Multimodal Learning

Mathematics Subject Classification

68T45

Explorando y Visualizando Comprensión de la Expresión Referente

por David Álvarez Rosa

Resumen

La interacción humano-máquina es uno de los objetivos principales actualmente en el ámbito de la Inteligencia Artifcial. En este trabajo se contribuirá a facilitar esta interacción explorando la novedosa tarea de Comprensión de la Expresión Referente (CER), consistente en: dada una expresión referente —que puede ser una frase lingüística o habla humana— y una imagen, detectar el objeto al que la expresión se refiere (i.e., conseguir una segmentación binaria del objeto referido). El caracter multimodal de este cometido hará necesario el uso de diferentes arquitecturas de aprendizaje profundo, entre ellas: redes neuronales convolucionales (visión artificial); y redes neuronales recurrentes y el modelo *Transformer* (procesamiento del lenguaje natural).

Esta tesis se presenta como un documento autosuficiente que puede ser entendido por un lector sin conocimientos previos en aprendizaje automático. El grueso del trabajo consiste en un estudio exhaustivo de la tarea de CER: desde las aplicaciones; hasta el estudio, comparación e implementación de modelos; pasando por una descripción completa del estado del arte actual. Así mismo, se presenta una página web funcional, gratuita y pública en la que se permite la interacción de una manera sencilla con el modelo descrito en este trabajo.

Palabras clave

Comprensión de la Expresión Referente

Inteligencia Artificial • Aprendizaje Automático • Aprendizaje Profundo

Visión Artificial • Porcesamiento del Lenguaje Natural

Aprendizaje Multimodal

Clasificación Matemática por Temas

68T45

Explorant i Visualitzant Comprensió de l'Expressió Referent

per David Álvarez Rosa

Resum

La interacció humà-màquina és un dels objectius principals actualment en l'àmbit de la Intel·ligència Artifcial. En aquest treball es contribuirà a facilitar aquesta interacció explorant la nova tasca de Comprensió de l'Expressió Referent (CER), que consisteix en: donada una expressió referent —que pot ser una frase lingüística o parla humana— i una imatge, detectar l'objecte a què l'expressió es refereix (i.e., aconseguir una segmentació binària de l'objecte referit). El caràcter multimodal d'aquesta comesa farà necessari l'ús de diferents arquitectures d'aprenentatge profund, entre elles: xarxes neuronals convolucionals (visió artificial); i xarxes neuronals recurrents i el model *Transformer* (processament de el llenguatge natural).

Aquesta tesi es presenta com un document autosuficient que pot ser entès per un lector sense coneixements previs en aprenentatge automàtic. El gruix de la feina consisteix en un estudi exhaustiu de la tasca de CER: des de les aplicacions; fins a l'estudi, comparació i implementació de models; passant per una descripció completa de l'estat de l'art actual. Així mateix, es presenta una pàgina web funcional, gratuïta i pública en la qual es permet la interacció d'una manera senzilla amb el model descrit en aquest treball.

Paraules Clau

Comprensió de l'Expressió Referent
Intel·ligència Artificial • Aprenentatge Automàtic • Aprenentatge Profund
Visió Artificial • Procesament del Llenguatge Natural
Aprenentatge Multimodal

Classificació Matemàtica per Temes

68T45

Acknowledgements

I would like to express my gratitude to Prof. Sanja Fidler for giving me the opportunity to carry out this project under her supervision and allowing me to be part of her laboratory and connect with its members. I also want to express my thanks to the entire Vector Institute staff for allowing me to use their computational resources, as well as for remotely assisting me with any technical problems that arose.

Likewise, I want to thank Prof. Xavier Giró for his work as liaison co-supervisor between Canada and Barcelona and for being part of the evaluation panel of this thesis.

Moreover, I want to express my gratitude to *Fundació Privada Cellex* and the Interdisciplinary Higher Education Centre. They have been the engines of my academic education and those that have allowed me to be part of this adventure of studying two official bachelors' degrees simultaneously in the Politechnical University of Catalonia. Within this great team I want to give special thanks to Toni Pascual for his management of the mobility stay and the complications arising from the COVID-19 pandemic. Thanks also to Miguel Ángel Barja—with whom I have been lucky to be his student—for his role as director of the center.

Finally, I want to thank my family and friends for their unconditional moral support without asking me too much *"When will you graduate?"*—or at least not very often.

David Álvarez Rosa
July 22, 2021

Table of Contents

List of Acronyms

The acronyms used in this thesis have been divided into two blocks: those considered primary and those with the names for the models mentioned in this document.

Primary

Acronyms that refer to the main topic of this thesis used in this work with the notation used, their corresponding description and page list.

Notation	Description	Page List
AI	Artificial Intelligence	1–3, 10, 69, 81
ANN	Artificial Neural Network	1, 9, 14, 16, 119
API	Application Programming Interface	3, 73, 76, 78, 81, 85, 96, 108, 112, 113
ARIA	Accessible Rich Internet Applications	72
ASPP	Atrous Spatial Pyramid Pooling	47
BCE	Balanced Cross Entropy	36
BPTT	Backpropagation Through Time	19
CE	Cross Entropy	35, 36, 38, 48, 51
CNN	Convolutional Neural Network	11, 14–16, 41, 45, 76
COCO	Common Objects in Context	32, 34, 69
CS	Computer Science	1, 53
CSS	Cascading Style Sheet	3, 72, 73, 76, 78, 107
CV	Computer Vision	2, 3, 10, 11, 42, 81
DC	Dice Coefficient	37

Notation	Description	Page List
DL	Dice Loss	37, 38, 48, 49
DL	Deep Learning	1, 2, 10, 20, 23, 26, 75, 76
DNC	Distance to the Nearest Cell	36
ECTS	European Credit Transfer and Accumulation System	76, 78
FL	Focal Loss	36
FNN	Feedforward Neural Network	11, 13, 14, 17, 21, 26
GIoU	Generalized Intersection over Union	37
GPU	Graphics Processing Unit	10, 79, 81
GRU	Gated Recurrent Unit	19
HTML	HyperText Markup Language	3, 76, 78
IoT	Internet of Things	3, 4
IoU	Intersection over Union	36–40, 49, 51, 52, 57, 58
JS	JavaScript	3, 72, 73, 76, 78, 108
LSTM	Long Short Term Memory	18, 41
mIoU	mean IoU	39, 40
ML	Machine Learning	1, 6, 9, 26–28, 38, 75, 76, 81, 120
MLM	Masked Language Model	47
NLP	Natural Language Processing	2, 3, 11, 16, 20, 42, 81
NSP	Next Sentence Prediction	47
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning	40
RE	Referring Expression	2–5, 31, 32, 34, 40–45, 47, 48, 57, 60, 62, 64, 67, 69–73, 82, 83

Notation	Description	Page List
REC	Referring Expression Comprehension	2, 3, 5, 6, 31, 32, 34, 37, 38, 40, 41, 43, 45, 53, 62, 64, 69, 72, 73, 76, 81, 82
ReLU	Rectified Linear Unit	16, 25, 119, 121
RGB	Red, Green and Blue	10, 15
RMSProp	Root Mean Square Propagation	24
RNN	Recurrent Neural Network	11, 16–20, 41, 47
SGD	Stochastic Gradient Descent	23, 24, 53
STT	Speech to Text	3, 6, 53, 54, 73
TI	Tversky Index	37, 49
UI	User Interface	69
UX	User Experience	70
W3C	World Wide Web Consortium	71, 72
WCE	Weighted Cross Entropy	36

Models

Acronyms for model names used in this work with the notation used, their corresponding description and page list.

Notation	Description	Page List
ASGN	Adversarial Semantic Guidance Network	57
BERT	Bidirectional Encoder Representations from Transformers	47, 60
BRINet	Bi-directional Relationship Inferring Network	57, 58, 60
CAC	Caption Aware Consistency	57, 60
CMAttErase	Cross Modal Attention guided Erasing	42, 60
CMPC	Cross-Modal Progressive Comprehension	57, 58, 60

Notation	Description	Page List
CMRE	Cross Modal Relationship Extractor	44
CMRIN	Cross Modal Relationship Inference Network	44
CMSA	Cross-Modal Self-Attention	57, 60
DGA	Dynamic Graph Attention Network	44
DMN	Dynamic Multimodal Network	57
FAOA	Fast and Accurate One-stage Approach	60
GGCN	Gated Graph Convolutional Network	44
LGRAN	Language Guided Graph Attention Network	43, 60
MAttNet	Modular Attention Network	42, 57, 60
MMI	Maximum Mutual Information	58, 60
NMTree	Neural Module Tree	60
RefVOS	Referring Expressions for Video Object Segmentation	45, 47, 51, 53, 57, 60, 81
RMI	Recurrent Multimodal Interaction	57, 60
RRN	Recurrent Refinement Network	57, 60
STEP	See-through-Text Embedding Pixelwise	57, 60
ViLBERT	Vision-and-Language BERT	60

List of Figures

List of Tables

Begin at the beginning, the King said gravely,
"and go on till you come to the end: then stop."
—Lewis Carroll
Alice in Wonderland

Chapter 1

Introduction

Artificial Intelligence (AI) is one of the branches of Computer Science (CS) that is most fashionable in recent years[1]. It has gained great importance mainly due to its applications in industry and everyday world, such as autonomous driving. It is an area of research in which, curiously, there is no precise definition universally accepted by the community of researchers and developers who work every day in the field of AI. Nilsson [Nil09][2], however, provides a useful definition.

> *Artificial intelligence is that activity devoted to making machines intelligent, and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment.*
>
> —Nilsson [Nil09]

γ

In other words, AI is a very broad concept that encompasses all those systems that perform tasks that can be considered intelligent. Within this broad world of AI is the scope of Machine Learning (ML), that is the set of computational algorithms that are responsible for automatically improving models through experience and with the use of *data*. ML algorithms build models using sample data to be able to make predictions or make decisions in future new situations without being explicitly programmed for it. That is, it seeks to imitate human learning, which is very interesting in various fields, such as automation for example.

Within this scope appear the Artificial Neural Networks (ANNs), which are the basis of a large family of ML methods called Deep Learning (DL). The adjective "deep" arises from the fact that neural models make use of multiple layers in the network. The models presented in this work will be part of precisely this area, as will be seen later (see Chapter 4 on page 45).

Furthermore, this work can be classified according to the type of data used, as a model of *multimodal learning*. Data can be of a different nature: images, text, audio,

[1] On July 22, 2021, current date of the document.

[2] Nils J. Nilsson. *The Quest for Artificial Intelligence*. 1st. USA: Cambridge University Press, 2009. isbn: 0521122937.

video, etc. This thesis will mix three of these types (audio → text and image). Next in Section 1.1 the work of this thesis will be described.

1.1 Description and Motivation

Referring Expression Comprehension (REC) is the task of, given a Referring Expression (RE)—is a linguistic phrase or human speech—and an image, generate a binary mask for the object which the phrase refers to. This type of task is framed within the field of multimodal learning: at the intersection between Computer Vision (CV) and Natural Language Processing (NLP).

(a) Man with cap **(b)** Laptop on the right **(c)** Army officer white suit

Figure 1.1. Examples of Referring Expression Comprehension. As you can see, we can refer to objects in the image with RE in natural language and segmentation occurs. Figures created by the author (all). View images in color to better appreciate segmentation.

In Figure 1.1 we can see some examples of this type of task. As we can see, the input will consist of two entities: one RE and an image. The model will be in charge of generating the segmentation of the object to which the phrase refers. We see that the RE shown are different types: relation of the object to segment with *other object* (man with a cap), type of object + relative *positioning* (laptop on the right) and object description + *color* distinctive (army officer white suit).

Optionally this task can be exploited and expanded in various ways. Among them, expanding the input and output set that the model can obtain. For the *input*, We can propose more general models that are capable of understanding RE from audio, without having to enter the phrase manually. Likewise, the model can be expanded accepting in addition to images, videos. In this thesis we will work on the part of audio. Furthermore, the *output* can also be extended by generating, in addition to the binary mask of the segmentation, a bounding box.

This work will add a great facility mainly in the human-computer interaction, so it is of great practical interest. Different applications of this model will be discussed in Section 1.2 on the next page.

1.1.1 Objectives

This research thesis has different objectives. The first one is to *learn* the operation of AI, without any prior knowledge. Mainly in the area of DL, which is where the presented model fits. It is essential to know the fundamentals of neural models to be able to understand in depth how it is possible to solve the problem of REC. This *learning* objective can be divided into learning the DL fundamentals and understanding the state-of-the-art papers for solving the task of REC.

With this knowledge, we will proceed to *modeling*, the search for models that work well in REC and in Speech to Text (STT). Its way of operation and how to improve it will be studied. For this, the Python programming language has been used with the PyTorch framework. All these results will be collected on an interactive website for *visualization*. Before working on web development, it will be necessary to $\sqrt{5}$
learn about front-end web programming languages (HTML, CSS, JS), languages to develop the Application Programming Interface (API) in the back end (PHP) and use of web servers (Apache).

Finally, there are the *academic* objectives related to this bachelor's thesis. These include the writing of this report, and the creation and preparation of the presentation.

1.2 Applications

The use of REC can have applications of various kinds. In recent years, robotics and home automation are gaining great importance. This work enhances the interaction between human and robot/computer. For example, it could make it easier for a machine to understand commands from a human. The possible applications of this work have been divided into four large groups: theoretical, industry, home automation and IoT, and security.

Theoretical

The creation and study of models in the field of multimodal learning using deep learning can end up having applications in different fields. Knowledge transfer between fields in AI is typical: many times CV and NLP end up sharing similar techniques.

In this specific case, we have precisely the interaction between models for language and for vision. This could be useful in the development of new models in the future in the field of multimodal learning. In addition, the creation of a website with which to interact with the different versions of the models, constitutes a tool for the evaluation of models. The general public is provided with a simple tool with which to perform multiple qualitative evaluations of the functioning of complex neural models.

e

Industry

In the industrial field, the model presented here could have applications in various fields. They could facilitate the interaction between operator/machine, thus improving the efficiency of a certain company and optimizing processes.

Among them could be that of the automotive world, as shown in Figure 1.2 on the following page, where several robotic arms operate on a vehicle being manufactured. Being able to refer to objects/parts of the vehicle using linguistic phrases would be very useful. For example, an operator could visually see that one of the welds is badly made and order the robotic arm to redo it with a RE of the type: lower right front door weld.

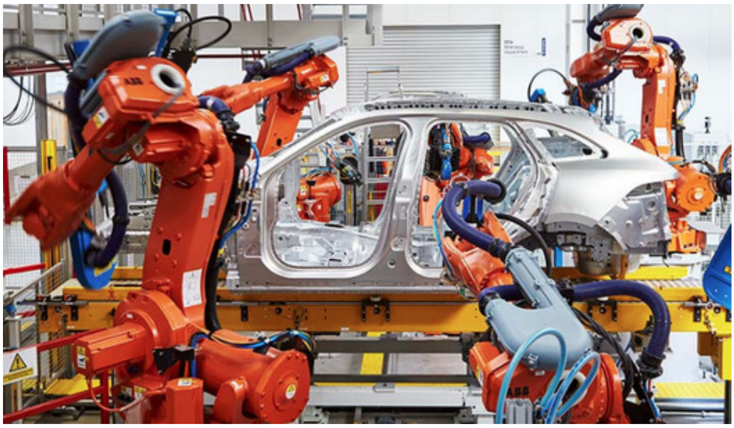


Figure 1.2. Robots in operation within an automobile factory. These processes are usually completely automated and controlled in real time. From *China's robot market is still No. 1*, by Asia Times Online [Asi21].

> **!** For some, there may be concern about whether this type of artificial intelligence applications could be detrimental to the professional future of society as a whole. It's an open debate, but, in the words of McKendrick [McK18], *"Artificial Intelligence Will Replace Tasks, Not Jobs"*.

This is just one example of the many applications that these models could have in the industry. The vast majority of sectors could benefit from the implementation of interaction systems between their operators and their machines by means of voice and using RE.

Home Automation and IoT

In the world of home automation and Internet of Things (IoT), so fashionable today, the topic that concerns this thesis could also be useful. Mainly for the ease it adds to the interaction between machines and humans.

In Figure 1.3 on the next page we can see some examples in which interaction in day-to-day tasks could be facilitated. In this figure, a robot is seen taking cubes of different colors and another robotic arm taking food and medicine from a shelf and stacking the selected ones in a box.

Not only because of the ease that this could add to the lives of many people, but also because of the added interest it would bring to people with disabilities. For example, a person with some type of physical disability (or an elderly person), might need help choosing products in a supermarket. In this case, a robotic arm could help him, and this work would serve as a link between the two and facilitate interaction. The person could refer to the products they want to purchase by simply using their voice and referring to it in *natural* language.

(a) Cubes of different colors **(b)** Food and medications

Figure 1.3. Examples of applications in robotics. They can facilitate many interactions and tasks of day to day. From *China Arduino Robot Arm*, by Alibaba Group Holding Limited [Ali21] (left) and from *The Future of Material Handling*, by IAM Robotics, LLC [IAM21] (right).

Security

Another possible application would be that used by the police to control road safety. In Figure 1.4 we can see one of these drones. Today, drones are already used for road control, but their control is done more manually. A much more automated approach is proposed here.

Figure 1.4. Image of one of the drones that can be employed by the authorities in order to guarantee road safety. From *Drone used to track suspects after Rte. 22 crash News*, by Pocono Record — Daily Newspaper [Poc19].

This type of drones could incorporate systems such as the one presented in this final degree project to be able to track vehicles using voice commands. For example, syntactic structures composed of action and RE could be used. Different *actions* could be desired to control and guarantee road safety such as: follow, record, speed up, etc. For the *REs* this would correspond to linguistic phrases that identify the vehicle to be studied or the offender to be pursued: blue/black/red/... car, large truck, sports car, van on the right, etc. In this way, combinations such as: "speed up the black car" or "record the sports car" could help control road safety. This work, as has already been commented previously, focuses on REC and not on the use made of this comprehension after.

1.3 Thesis Overview

Here a description of the different chapters that make up this thesis will be presented.

Chapter 1 It is this chapter and it is an introductory chapter at a general level of the subject that will be dealt with in this thesis. The motivation behind this work, its objectives and its possible applications will be discussed. This Chapter, entitled *Introduction*, begins on page 1.

Chapter 2 This chapter will focus on the general theoretical foundations in the field of ML, so that the following chapters can be understood. This Chapter, entitled *Theoretical Background*, begins on page 9.

Chapter 3 It will deal with the central theme of this thesis. The problem will be formulated in a concrete way, the existing datasets and the evaluation techniques for this task will be presented and the existing state-of-the-art models will be discussed. This Chapter, entitled *Referring Expression Comprehension*, begins on page 31.

Chapter 4 This chapter will introduce the concrete models used both for the task of REC and for STT. It will be discussed how it has been trained, the different versions that have been studied (model iterations) and how it has behaved. This Chapter, entitled *Models*, begins on page 45.

Chapter 5 For the selected model, an evaluation will be made—quantitative and qualitative—of the results. Likewise, this model will be compared with current state-of-the-art works. This Chapter, entitled *Results and Comparison*, begins on page 57.

Chapter 6 It will present all the code developed to achieve an interactive web application with which it will be possible to evaluate and validate the model in a simple way. Both front end and back end will be discussed. This Chapter, entitled *Visualization*, begins on page 69.

Chapter 7 In this chapter the project will be analyzed from a management point of view. The different activities carried out and its programming will be summarized. Finally, an analysis of economic cost and environmental impact will be made. This Chapter, entitled *Project Analysis*, begins on page 75.

Chapter 8 The thesis will be briefly summarized, the results obtained will be discussed, a global conclusion will be presented and future lines of research will be provided. This Chapter, entitled *Conclusions*, begins on page 81.

Supplementary material will also be included in the appendices.

Appendix A All files created and used in this project will be displayed graphically. This Appendix, entitled *File Structure*, begins on page 85.

Appendix B This appendix shows the details of the implementation of the models, and of the web. The most representative code files are shown. This Appendix, entitled *Implementation Details*, begins on page 89.

Appendix C Extra material that has been referenced in this thesis will be added here. This Appendix, entitled *Supplementary Material*, begins on page 119.

Finally, after the appendices, there is the full *Bibliography* (begins on page 123) and an *Alphabetical Index* to facilitate the search for terms and topics (begins on page 133).

How to Read this Thesis

Throughout the thesis, four types of different boxes will be used to include extra content: *quote* box, *example* box, *remark* box and *code* box.

> *This is a quote box.*
>
> —Name SURNAME

This is an example box.

! This is a remark/alert box.

```python
def code_box():
    """This is a code box."""
    return None
```

These different colored boxes also have different shapes to ensure the reading of the printed document in black and white, and to guarantee accessibility for the color blind.

Without theory, there is nothing to revise.
Without theory, experience has no meaning.
Without theory, one has no questions to ask.
Hence, without theory, there is no learning.
—William Edwards

Chapter 2

Theoretical Background

Neural networks, more properly referred to as Artificial Neural Networks (ANNs) are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Dr. Robert Hecht-Nielsen (inventor of one of the first neurocomputers), in an article by Caudill [Cau87][1], defines a neural network as:

> *. . . a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*
>
> —Caudill [Cau87]

Although the analogy made above of an ANN with a biological brain, there is no need for this, we can just think of a neural network—from a mathematical perspective—as an optimization problem. We can think of the whole network to be a function that takes some inputs to some outputs, and this function dependent on parameters. The idea is to adjust this parameters to get a function that works well with some known dataset, and we will trust that it will generalize well. If the network is big enough and we carefully adjust the parameters, we will be able to learn and compute very complex functions.

This chapter will consist of establishing the theoretical foundations that will later be useful to understand the models that will be presented. The reader will be assumed to have a base of mathematical foundations, but not necessarily with knowledge in the field of Machine Learning (ML). In this way, we will begin by laying the foundations on the use of the mathematical tool of tensors in the Section 2.1 on the next page and the main existing architectures of neural networks will be defined (see Section 2.2 on the following page). The two main paradigms of ML, training (see Section 2.3 on page 21) and testing (see Section 2.4 on page 29) will be discussed below. In this way, this chapter will contain all the necessary knowledge to develop neural models in a general view.

[1] Maureen Caudill. "Neural Networks Primer, Part I". in: *AI Expert* 2.12 (Dec. 1987), pp. 46–52. issn: 0888-3785.

2.1 Tensors

In our context, it will be useful to use the mathematical tool of tensors. There are different approaches to define this mathematical objects, among then, defining them as multi-dimensional arrays of real numbers, as follows: a *tensor* is an element $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_r}$, with $n_1, \ldots, n_r \in \mathbb{N}$. The number r is called the *rank* of the tensor.

Similarly to real vectors from $\mathbb{R}^n$, each of the elements of a tensor $\mathbf{T}$ can be referred to using a multi-index $i = (i_1, \ldots, i_r) \in \mathbb{N}^r$. The notation $\mathbf{T}_i$, or $\mathbf{T}_{i_1,\ldots,i_r}$, refers to the element indexed by i of the tensor $\mathbf{T}$.

> Examples of tensors include scalars (view as tensors of rank 0), vectors (rank 1 tensors) and matrices (rank 2).

Most popular Deep Learning (DL) frameworks such as PyTorch—used in this thesis—are just programming libraries that provide efficient tensor data structures and fast tensor computation—with the ability to compute in Graphics Processing Unit (GPU). These tensors store the data used to train neuronal models. In the case of Computer Vision (CV), the following tensors appear:

- **RGB image.** A RGB image can be interpreted as a tensor of rank 3, $\mathbf{I} \in \mathbb{R}^{C \times H \times W}$, where C corresponded to the number of channels (i.e., in this case, $C = 3$), H corresponds to the height of the image and W to its width.[2]
- **Batch of RGB images.** A batch of RGB images is a set of RGB images, therefore, can be interpreted as a tensor of rank 4, i.e., $\mathbf{I} \in \mathbb{R}^{B \times C \times H \times W}$, where B corresponds to the batch size, and (C, H, W) have the same meaning as in the case of an RGB image.

2.1.1 Tensor Operations

Tensors or rank 1 or 2 correspond to real vectors and real matrices, therefore, all basic operations from linear algebra apply (e.g., vector sum, matrix-vector multiplication). For tensor of higher rank, it is useful define element-wise operations for any binary operation from real numbers.

Let $\mathbf{T}$ and $\mathbf{S}$ be tensors of rank r. Let $\star$ be a binary operation for real numbers (e.g., ordinary sum or multiplication). Then, the element-wise operation is defined as follows,

$$(\mathbf{T} \star \mathbf{S})_i = \mathbf{T}_i \star \mathbf{S}_i, \tag{2.1}$$

for all multi-indices $i \in \mathbb{N}^r$.

2.2 Neural Network Architectures

During the last few years, there have been significant advances in the field of Artificial Intelligence (AI) and multiple new models of neural networks have appeared. In a

[2] Gray scale images also fall into this category with just 1 channel (i.e., using the above notation $C = 1$).

very general classification of neural models we can find different ones.

- **Feedforward Neural Network.** These networks are the simplest type of neural model that exists and on which more complex models are built. A more detailed description of this type of network can be found in Section 2.2.1.
- **Convolutional Neural Network.** This type of neural network is especially useful in image processing, since it allows preserving the spatial information of the relationship between pixels. They will be studied in Section 2.2.2 on page 14.
- **Recurrent Neural Network.** Ideal neural models for the analysis of time series (such as text or audio). They will be studied in Section 2.2.3 on page 16.
- **Transformers.** The transformer-based models are part of the current[3] state of the art. They are being used extensively in Natural Language Processing (NLP) and studying their possible applications in CV and image processing. They will be described in Section 2.2.4 on page 20.

Of course other types of architectures exist in addition to those presented in this work. However, understanding these it is easier to extrapolate to other models, since many are based on these. In addition, in the multimodal learning environment, many times what are used are combinations of the existing ones.

2.2.1 Feedforward Neural Network

Feedforward Neural Networks (FNNs) are the simplest type of neural models that exists and that constitute the basis for more complex neural structures. This type of neural network is characterized by a neural structure organized in different layers, so that the neurons between adjacent layers are connected to each other by arcs. An example of this type of architecture is shown in Figure 2.1 on the next page.

This type of neural network has 3 main constituent parts, which will be discussed below; these are: layers, neurons and connections.

Layers

The layers are just a collection of neurons, we will distinguish between three types depending on its position in the network: *input* layer (patterns are presented to the network via this layer), *hidden* layer (all the inner layers) and *output* layer (is the last layer, where the answer is obtained).

We will denote with L the number of layers and with n_l the size of the l-th layer. This number of layers and the number of neurons in each is a hyperparameter that is not easy to set, it is an area of active research to find the optimal and most efficient topologies for a given problem.

[3] On July 22, 2021, current date of the document.

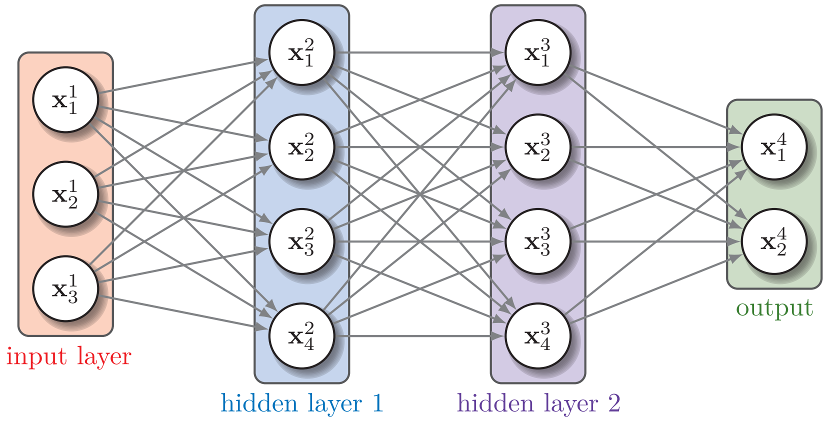


Figure 2.1. Example of a Feedforward Neural Network. It consists of 4 differentiated layers of neurons and the connections between them are shown graphically with arrows. Figure created by the author.

Neurons

Neurons are the core component of any neural network. Basically there are three subparts that form a neuron.

- **Value.** Each neuron holds a value, it will be denoted by $x_i^l \in \mathbb{R}$ for the i-th neuron in the l-th layer. Of course, it should be satisfied $1 \leq i \leq n_l$. We will use the notation $\mathbf{x}^l$ for the vector of all the values in the l-th level. When we speak of the input vector, we may omit the superindex, i.e., we will use $\mathbf{x}$ to denote $\mathbf{x}^0$. Similarly, for the output layer, we will use $\hat{\mathbf{y}}$ to refer to $\mathbf{x}^L$.

- **Bias.** Also each neuron has a bias, denoted as b_i^l for the i-th neuron in the l-th layer. Is then true that $1 \leq i \leq n_l$. The vector of all biases in the l-th layer will be denoted by $\mathbf{b}^l$.

- **Activation function.** All neurons have an activation function $f_i^l \in \mathcal{C}^1(\mathbb{R}, \mathbb{R})$ for the i-th neuron in the l-th layer.[4] Of course, it is needed $1 \leq i \leq n_l$. The regularity assummed for this functions is important, since we will be optimizing in the future by taking derivatives.See Appendix C.1 on page 119 for examples of these functions.

Connections

As we discussed, in the topology of this network, all neurons between adjacent layers are required to be connected—with the so-called *connections*—that should have associated a *weight*. For the connection between the i-th neuron in the l-th layer and

[4] Usually all the activation functions are neuron-independent (i.e., f_i^l does not really depend on i or l).

the j-th neuron in the $l+1$-th layer, we will denote this weight by $w_{ij}^l \in \mathbb{R}$. The set of all this weights is as follows,

$$\{w_{ij}^l \mid 1 \leq i \leq n_l,\ 1 \leq j \leq n_{l+1},\ 1 \leq l < L\} \subset \mathbb{R}. \tag{2.2}$$

The *matrix* of all weights in the l-th layer will be denoted by $\mathbf{W}^l$. This is, $(\mathbf{W}^l)_{ij} = w_{ij}^l$.

To gain some intuition on how this work, let's think about the handwritten recognition problem. Suppose we have a set of images with handwritten digits in it and that we will like to implement an ANN that is capable of recognizing that digits.

In this case, if the digits images are of 28×28 pixels, the input layer will consist of 784 (28×28) neurons and each neuron will hold the gray scale value of a pixel. For the output layer we will need 10 neurons (one for each number between 0 and 9), and we will like that when we feed our network with an image holding a handwritten number 3, then the output is one 1 in the position corresponding to the number 3 and the rest of zeros. In this case we want the output vector to be really a *probability* vector, where higher probability indicates greater similarity with that number

Model Feed Forward

From now on let's suppose we are working with a FNN with different layers and we will be using the same notation used before. The values of the neurons can be computed with

$$x_i^l = f_i^l \left(\sum_{k=1}^{n_{l-1}} w_{ik}^{l-1} x_k^{l-1} + b_i^l \right). \tag{2.3}$$

This formula is sometimes referred to as the feed-forward formula or forward propagation. It's important to note that it's a recursive formula, once the values the neurons in the input layer are known, we can iterate computing the values of the neurons in the next adjacent layer, until we reach the output layer. In this network we can think of information traveling in one direction, forward, from the input layer, through the hidden layers to the output neurons.

Training Backpropagation

In order to be able to train our neural network, it is mandatory to define an error function (also known as loss function) that quantifies how good or bad the neural network is performing when fed with a particular dataset. We will use the below notation.

- **Dataset.** Will be denoted by and consists of input-output pairs $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ represents the input and $\mathbf{y}$ the *desired* output. We shall denote the size (cardinal) of the dataset by N. Of course, in terms of our ANN x corresponds to the values of the first (or input) layer and $\mathbf{y}$ to the output (or last) layer.

- **Parameters.** Parameters of our ANN are both the connection weights w_{ij}^l and the biases b_i^l, we will denote the set of all parameters by $\boldsymbol{\theta}$. Keep in mind that $\boldsymbol{\theta}$ is just a set of real vectors of $\mathbb{R}^D$ where D denotes the number of weights and biases.

The error function quantifies how different is the desired output $\mathbf{y}$ and the calculated (*predicted*) output $\hat{\mathbf{y}}$ of the neural network on input $\mathbf{x}$ for a set of input-output pairs $(\mathbf{x}, \mathbf{y}) \in \Omega$ and a particular value of the parameters θ. We will denote the error function by $E(\Omega, \theta)$ and we will assume that it is continuously differentiable (i.e., $\mathcal{C}^1$). The training process itself will be discussed in more depth in Section 2.3 on page 21.

2.2.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a class of ANNs that are widely used in computer vision. They are commonly applied in image processing, since they have the main characteristic of having the ability to preserve the spatial relationship between pixels. Likewise, they are regularized versions of FNN (explained in Section 2.2.1 on page 11), that is, it is less likely to overfit the data.

LeCun [LeC98][5], a renowned scientist known as the father of CNNs, defines them in a similar way to other neural models—since they are trained in the same way—and with the only difference present in their topology.

> *CNNs are a special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm. Where they differ is in the architecture.*
>
> —LeCun [LeC98]

Analogously to the case of fully connected networks, a CNN consists of an input layer, hidden layers and an output layer. In a CNN the hidden layers include layers that calculate convolutions. These convolutional layers are followed by different ones, among which the pooling layers, fully connected layers and normalization layers stand out. An example of this type of architecture can be seen in Figure 2.2 on the next page.

Convolutional Layers

The name comes from the fact that these layers use the mathematical operation of convolution[6]. This operation is defined (in its discrete version) as follows: given a pair of functions f, g defined on the set of integers $\mathbb{Z}$, the discrete convolution between f and g is given by,

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m], \tag{2.4}$$

[5] Yann LeCun. *LeNet-5, Convolutional Neural Networks — Personal Website*. http://yann.lecun.com/exdb/lenet/. [Online; accessed 15 February of 2021]. 1998.

[6] Full link for "convolution": https://en.wikipedia.org/wiki/Convolution

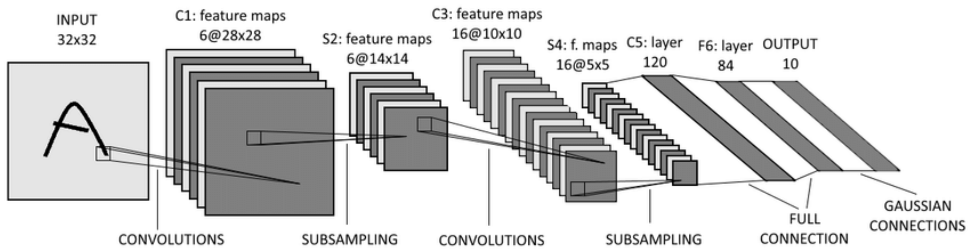


Figure 2.2. Example of topology of a CNN. Specifically, it is LeNet-5, a CNNs created by Yann LeCun. From *LeNet-5, Convolutional Neural Networks — Personal Website*, by LeCun [LeC98].

where $\star$ is the convolution operator.

This is the mathematical definition of the discrete convolution. However, the use of convolution in image processing does not work exactly as defined in Equation (2.4) on the facing page. In practice, the input images[7] are three-dimensional tensors for the images RGB. In convolutional layers, a two-dimensional convolution is made with a three-dimensional filter (also called kernel) on each channel of the input image and then all these feature maps are stacked in the output tensor, where the number of output channels coincides with the number of filters in that layer.

We can mathematically express the operation performed on the convolutional layers as follows. Let $\mathbf{X}$ be the input characteristics map, $\mathbf{Y}$ the output characteristics map and the filter $\mathbf{F}$. The convolution is then defined as follows,

$$\mathbf{Y}_{i,j,k} = \sum_{l,m,n} \mathbf{X}_{l,j+m,k+n} \mathbf{F}_{i,l,m,n}, \tag{2.5}$$

where the sum is performed for all valid l, m, n indices (this will depend on *padding*[8] of the input image).

An example of convolution is shown graphically in Figure 2.3 on the next page. As can be seen for each of the elements of the output tensor, the computation of Equation (2.5) is performed with the filter shown.

Pooling Layers

Another type of layer used in this type of network is pooling. Pooling layers reduce the dimension of the network by combining the output of neurons in one layer into a single neuron in the next layer. These types of layers also add non-linearities to the model. They also make CNNs less sensitive to small local changes in spatial location.

There are different types of pooling. The best known are *max* pooling and *average* pooling. The first uses the maximum value of each cluster of neurons in the previous

[7] At deeper levels in a CNN we will stop understanding the layers as images and call them *feature maps*.

[8] There are different techniques to define the padding of an image. The most typical is known as zero-padding, which consists of adding 0 vectors around the image until it is complete in order to carry out the desired convolutions.

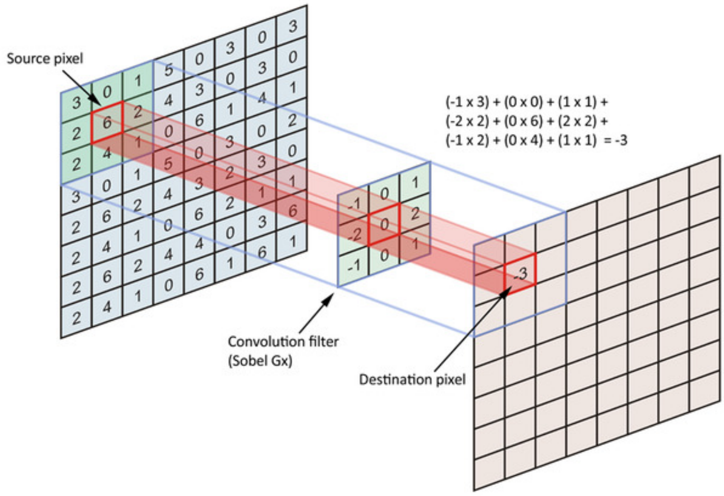


Figure 2.3. Example of a convolution with a kernel (filter) of dimensions $(3, 3, 1)$ and an input feature map (tensor) of size $(8, 8, 1)$. From "Graduation Thesis Implementing and Optimizing Neural Networks using Tiramisu", by HACHILIF et al. [HBB19].

level. Average pooling, however, uses the average value. In Figure 2.4 on the next page we can see a representation of a 2×2 max-pool. In this specific case, the mathematical operation to be performed is given by the following expression,

$$f_{X,Y}(S) = \max_{a,b=0}^{1} S_{2X+a,2Y+b}, \tag{2.6}$$

where S is every depth slice in the input. This discharges 75% of the activations.

Activation Functions

Analogously to fully connected neural networks, in this case it is also common to use activation functions with the aim of introducing non-linearities in the model without affecting the receptive fields of the convolutions. They are typically added just after convolution.

The main functions used are the following: Rectified Linear Unit (ReLU), hyperbolic tangent, sigmoid function, etc. See Appendix C.1 on page 119 for a graphic representantion of these functions.

2.2.3 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a class of ANNs that are widely used with temporal sequences (as for example in the scope of NLP). Analogously to the CNNs that are suitable for image processing, the RNNs are a type of neural network specialized to process sequences of values of the type $\mathbf{x}^1, \ldots \mathbf{x}^\tau$ with $\mathbf{x}^t \in \mathbb{R}^n$.

Looking at Figure 2.5 on the next page, we can see the architecture of a RNN. The input values $\mathbf{x}^1, \ldots \mathbf{x}^\tau$ are fed into the network in an orderly manner. Therefore,

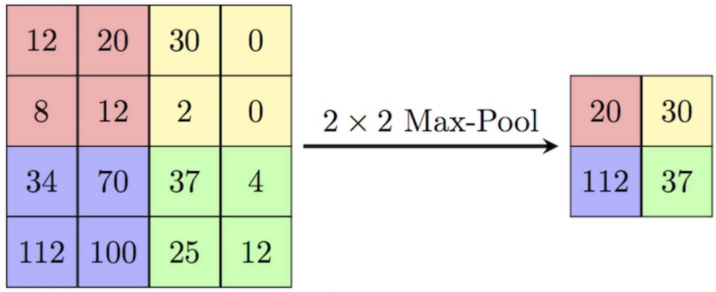


Figure 2.4. Pooling layer example with 2×2 max pooling. The input tensor is divided into blocks 2×2 and the Max Pool returns an output tensor with the maximum value of each block. From *Max-pooling/Pooling — Computer Science Wiki*, by COMPUTER SCIENCE WIKI [Com18].

to obtain the output vector $\mathbf{y}^t$, the hidden state $\mathbf{h}^{t-1}$ and the input $\mathbf{y}^t$ are necessary. Prior inputs $\mathbf{x}^1, \dots \mathbf{x}^{t-1}$ are represented by the hidden state $\mathbf{h}^{t-1}$, therefore the output $\mathbf{y}^t$, depends exactly of all the inputs until time t.

(a) Basic RNN **(b)** Unfolded basic RNN

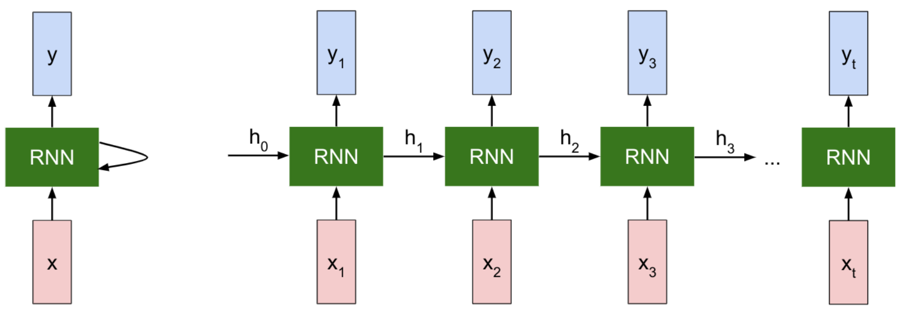


Figure 2.5. Basic topology of a Recurrent Neural Network (RNN). Two versions of the same network are shown, one is the compact version and the other is the same network but unfolded in time. From *CS231n: Convolutional Neural Networks for Visual Recognition*, by LI et al. [LKX20].

Different Types of RNN

In oposition to FNNs, that send one input to one output[9], RNNs do not. There are different possibilities, in which the length of the inlet and outlet can vary. All the different types of architectures that may exist are listed in Figure 2.6 on the following page.

These different possibilities then have their application in various fields such as: image captioning (`one to many`), action prediction (`many to one`), video captioning

[9] The input and output can be vector. It does not mean, therefore, that they are only a number, it means that they are only *one* multidimensional vector of $\mathbb{R}^n$.

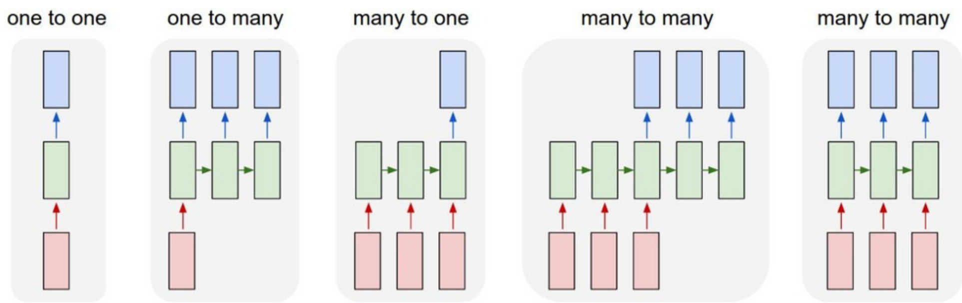


Figure 2.6. Types of architectures for RNN. All possibilities are considered (i.e., all combinations are presented). From *CS231n: Convolutional Neural Networks for Visual Recognition*, by LI et al. [LKX20].

(many to many first option), and video classification on frame label (many to many second option). The extreme case of one to one corresponds precisely to a normal fully connected layered neural network such as the one studied in Section 2.2.1 on page 11.

Model Feed Forward

We can compute the value of the neurons with a recurring formula, involving the hidden states $\mathbf{h}^t$ and the values of the input vectors $\mathbf{x}^t$. In the simplest case of a *simple* RNN, this computation consists of,

$$\mathbf{h}^t = f_W(\mathbf{h}^{t-1}, \mathbf{x}^t), \tag{2.7}$$

and then the output vector $\mathbf{y}^t$ can be calculated using the following expression,

$$\mathbf{y}^t = W_{hy}\mathbf{h}^t, \tag{2.8}$$

where W_{hy} is an array of parameters (trainable).

Variant RNN Architectures

The model defined above is the simplest version for a RNN. The computation of the output values with the Equations (2.7) and (2.8) can cause problems with the gradients and, therefore, limit the ability of the information to travel in time.[10] We introduce two new variants to solve these problems.

- **Long Short Term Memory (LSTM).** A unit of LSTM is composed of different gates that define its behavior. Its name is given by the function they perform: *input* gate (i), *output* gate (o) and *forget* gate (f). A graphical representation of this type of elements is shown in Figure 2.7 on the next page.

[10] These problems are know as vanishing/exploding gradient. The problem of *exploding* gradients can be solved by gradient clipping (scaling it if the norm is too big). However for the *vanishing* gradient problem, it is necessary to change the RNN architecture.

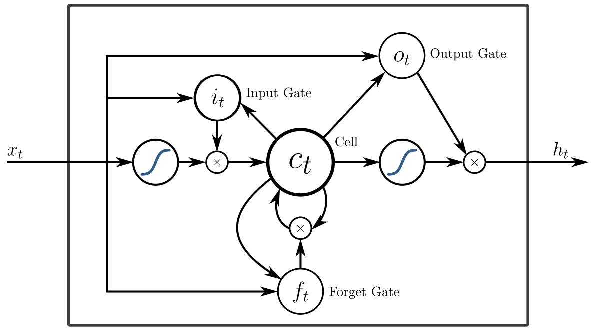


Figure 2.7. Long Short Term Memory (LSTM) representation. Input (i), output (o) and forget (f) gates are shown in the image. From *Long short-term memory — Wikipedia, The Free Encyclopedia*, by Wikipedia contributors [Wik21b].

The value of the different gates is described by the following equation,

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h^{t-1} \\ x^t \end{pmatrix}. \tag{2.9}$$

And then, the value of the cell and the hidden state can be computed with,

$$\begin{cases} c_t &= f \odot c_{t-1} + i \odot g, \\ h_t &= o \odot \tanh c_t. \end{cases} \tag{2.10}$$

The operator $\odot$ is the element wise multiplication known as Hadamard product.

- **Gated Recurrent Unit (GRU).** They are another type of gating mechanism introduced by Cho et al. [Cho+14][11]. Its operation is governed by the following system of equations,

$$\begin{cases} z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z), \\ r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r), \\ \hat{h}_t &= \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t. \end{cases} \tag{2.11}$$

Where $\hat{h}_t$ is known as the candidate activation vector, z_t is the update gate vector and r_t is the reset gate vector. The activation functions σ_g correspond to the sigmoid function and ϕ_h is the hyperbolic tangent. Other activation functions would also be possible.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv preprint arXiv:1406.1078* (2014). arXiv: `1406.1078 [cs.CL]`.

Trainning: Backpropagation Through Time

Analogously to other models, here the training of RNN is also carried out using first-order optimization methods, i.e., calculating the partial derivatives of the error function with respect to the model parameters. In this specific case of RNN, the backpropagation process is known as Backpropagation Through Time (BPTT), which is a generalization of backpropagation in feed-forward networks.

2.2.4 Transformer Model

The Transformer is a DL model recently introduced by VASWANI et al. [Vas+17][12]. Here they presented the idea that recurrent building blocks are not needed in a model to work well in NLP tasks. Its authors define it precisely.

> *The Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.*
>
> —VASWANI et al. [Vas+17]

They propose a new architecture that is capable of maintaining an *attention mechanism* while processing temporal sequences in parallel: the entire sequence as a whole instead of going element by element. This would improve RNN models in which training is sequential. The attention function that this model uses in the so-called "Scaled Dot-Product Attention", which works as follows: given *queries* (Q), *keys* (K) and *values* (V), the attention is computed as (in matrix notation),

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.12}$$

where d_k is the dimension of the queries and keys.

The problem with this attention function is that it only allows one way that words can interact with each other. To solve this, they propose a "Multi-Headed Attention", where h attention layers are used running in parallel (they use $h = 8$ in their work) and then concatenate the outputs (see Figure 2.8 on the facing page). This attention is computed as follows,

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O, \tag{2.13}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^k, VW_i^V)$ (see Equation (2.12)), and the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^k \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Currently there are different types of architectures based on this idea. The original architecture of the Transformer model is shown in Figure 2.9 on page 22. As you can see, two distinct segments can be distinguished:

[12] Ashish VASWANI, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, et al. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., June 2017, pp. 6000–6010.

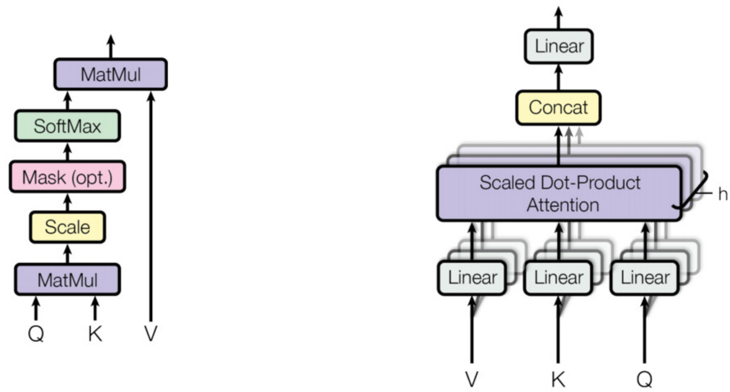


Figure 2.8. Transformer attention mechanism. Scaled Dot-Product Attention (right) and Multi-Headed Attention that involves attention layers running in parallel (left). From "Attention is All You Need", by VASWANI et al. [Vas+17].

- **Encoder segment.** It takes the inputs, generates an embedding of them, encodes the positions, computes where each word has to attend to in a multi-context setting and then outputs a new intermediate representation.
- **Decoder segment.** Take the entries in the target language, generate an embedding for them with encoded positions, calculate in which each word has to attend, and then combine the output of the encoder with the output so far. The result is a prediction for the next token.

This model uses input/output embeddings to obtain vector representations that can be fed into the Transformer. And, at the end, after the last segment (see Figure 2.9 on the next page), the encoded output is fed into a FNN (is a linear transformation, no activation functions presented) and into a softmax layer to obtain a vector of *probabilities*.

2.3 Training

The process of trainning a neural networks consists in adjust the parameters of the model to fit a particular dataset. Moreover, it is mandatory to define a loss function function (also known as error function) that quantifies how good or bad the neural network is performing when fed with a dataset.

A dataset Ω consists of input-output pairs (x, y), where x represents the input and y the *desired* output. We shall denote the size (cardinal) of the dataset by N.

The loss function quantifies how different is the desired output y and the calculated (*predicted*) output $\hat{y}$ of the neural network on input x for a set of input-output pairs $(x, y) \in \Omega$ and a particular value of the model parameters θ. We will denote the loss function by $\mathcal{L}(\Omega, \theta)$ and we will assume that it is continuously differentiable (i.e., $\mathcal{C}^1$).[13]

[13] The regularity assumed for this functions is important, since we will be optimizing in the future by

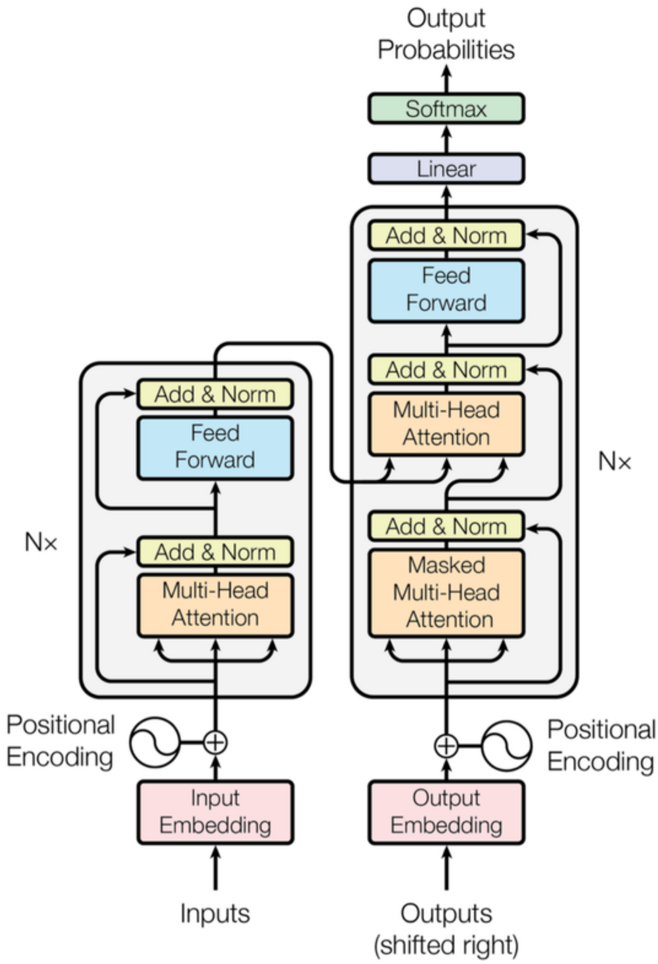


Figure 2.9. Transformer model architecture/topology. Here you can observe the two main segments: the encoder and the decoder. From "Attention is All You Need", by Vaswani et al. [Vas+17].

It is common (and we will assume it that way) that the loss function is a mean of the errors of a particular pair $(x, y) \in \Omega$. This is, there exists a continuously differentiable function $\ell(x, y, \Omega)$, such that,

$$\mathcal{L}(\Omega, \theta) = \frac{1}{N} \sum_{(x,y)\in\Omega} \ell(x, y, \theta). \tag{2.14}$$

Now, what we will want to do is to optimize (minimize) this loss function in θ. This is, given a dataset Ω, we will want to approximate,

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\Omega, \theta), \tag{2.15}$$

given that the above exists.

2.3.1 Optimization

There exists several optimization techniques, among then, the most important, are gradient-based optimization algorithms. This techniques are iterative methods that, given an initial value $\theta^{(0)}$ for the parameters proceed, as follows,

$$\theta^{(t+1)} = \theta^{(t)} + \alpha\, \Delta\theta^{(t)}, \tag{2.16}$$

where α is called the step size (or learning rate), and $\Delta\theta^{(t)}$ is the weight update in step t. The goal is to find values for the parameters such that the loss function corresponds to a (global) minimum.

However, obtaining a global minimum is a very hard task, so a *local* minimum will be enough. It is a well-known result from multivariable calculus that if $\hat{\theta}$ is a local minimum of $\mathcal{L}$, then $\nabla\mathcal{L}(\hat{\theta}) = 0$, therefore the optimization methods will focus in finding stationary points (that, hopefully, correspond to global or local minimums).[14]

Optimization Methods

Different optimization methods exist, the most basic and best known method of first-order optimization is *gradient descent*. The idea is to move in the opposite direction of the gradient, that is, it is an iterative method, following the same rules as in Equation (2.16), consisting of the following,

$$\Delta\theta^{(t)} = -\nabla\mathcal{L}(\theta^{(t)}). \tag{2.17}$$

Considering the typical form of the loss function (see Equation (2.14)), the gradient can be computed using the derivative property of linearity,

$$\nabla_\theta\, \mathcal{L}(\Omega, \theta) = \frac{1}{N} \sum_{(x,y)\in\Omega} \nabla_\theta\, \ell(x, y, \theta). \tag{2.18}$$

computing partial derivatives (as discussed in Section 2.3.1).

[14] These are called first-order optimization methods since they only focus in first-order (partial) derivatives of the loss function. Higher order methods exists, but involve computing the Hessian of $\mathcal{L}$ and in the scope of DL it is too expensive in terms of computing resources.

Typically the datasets Ω used for training have very large cardinal ($N > 10^4$), so the computation of Equation (2.18) on the previous page is too expensive in terms of computational resources. Therefore, in practice, the alternative known as Stochastic Gradient Descent (SGD) is used, which consists of estimate the gradient at each iteration in a random subset from the dataset.

SGD is an implementation of Equation (2.17) on the preceding page in which it is estimated $\nabla\mathcal{L}(\theta^{(t)})$ using a randomly chosen subset $B \subset \Omega$. That is, the computation of Equation (2.18) on the previous page is replace with the following estimate,

$$\nabla_\theta\, \mathcal{L}(\Omega, \theta) \approx \frac{1}{|B|} \sum_{(x,y)\in B} \nabla_\theta\, \ell(x, y, \theta). \tag{2.19}$$

It is common to abuse the notation and denote by B the cardinal of the subset, and call it *batch* size.

Other optimization methods exist, mainly due to the presence of saddle points, which would stop the previous iterative methods since the gradient would cancel out. The best known are the following,

- **Momentum.** It consists of performing the following computation,

$$\Delta\theta^{(t)} = -\beta\Delta\theta^{(t-1)} - \nabla\mathcal{L}(\theta^{(t)}), \tag{2.20}$$

  where β is a hyperparameter. This algorithm maintains the previous velocity as an average of the above gradients and the parameter β can be understood as a parameter of "friction" (thinking in physical terms). This optimization process can be understood in physical terms as follows: supposing that the function to be optimized were two variables (parameters), the path we want to follow to find the minimum is the same as a ball dropped in the initial guess would follow (pulled solely by gravity and braked by friction with the ground). So, moment will help us avoid saddle points, which is very interesting in an optimization process.

- **Nesterov Momentum.** Similarly to previous method uses a term of "velocity", but calculating the gradient not at the current point, but at the point where the velocity would carry, i.e.,

$$\Delta\theta^{(t)} = -\beta\Delta\theta^{(t-1)} - \nabla\mathcal{L}(\theta^{(t)} + \beta\Delta\theta^{(t-1)}). \tag{2.21}$$

- **Root Mean Square Propagation (RMSProp).** This optimization algorithm includes an adaptive learning rate (decaying exponentially with the mean of the square of the gradients). The following update value is used,

$$\Delta\theta^{(t)} = -\frac{\mu}{\sqrt{\delta + v^{(t)}}}\nabla\mathcal{L}(\theta^{(t)}), \tag{2.22}$$

  where,

$$v^{(t)} = \rho v^{(t-1)} + (1-\rho)\left(\nabla\mathcal{L}(\theta^{(t)})\right)^2, \tag{2.23}$$

  ρ is the forgetting factor and δ is a small real number to make sure no divisions are made by 0.

- **Adaptive moment estimation (Adam).** Proposed by KINGMA et al. [KB14][15], in this optimization algorithm the following estimates are used,

$$\begin{cases} m^{(t+1)} = \beta_1 m^{(t)} + (1-\beta_1)\nabla\mathcal{L}(\theta^{(t)}), \\ v^{(t+1)} = \beta_2 v^{(t)} + (1-\beta_2)\big(\nabla\mathcal{L}(\theta^{(t)})\big)^2, \\ \hat{m} = \dfrac{m^{(t+1)}}{1-\beta_1^{t+1}}, \\ \Delta\theta^{(t)} = \dfrac{v^{(t+1)}}{1-\beta_2^{t+1}}, \end{cases} \tag{2.24}$$

  where δ is a small real number to make sure no divisions are made by 0 and β_1 and β_2 are the forgetting factors for gradients and second moment gradients. A good starting point for these parameters is usually set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The Adam method can be interpreted as that the first moment is taking the role of impulse and with the second it is used to adapt the learning rate. This is the algorithm that that we will be using for the training of models.

Weight Initialization

For every iterative method, an initial guess for is necessary $\theta^{(0)}$. In practice what you do is initialize the weights using a probability distribution, such as $U[-\sigma,\sigma]$ or $\mathcal{N}(0,\sigma^2)$. The value of σ can be chosen arbitrarily, although the one known as *Xavier initialization*, by GLOROT et al. [GB10][16], consisting of choosing the variance as the square root of the input dimension D_{in}.[17] This way we get activations are nicely scaled in all layers.

However, in Xavier initialization it assumes that the activation function is centered at 0, therefore, when the activation function ReLU is used, it is preferable to use another type of weight initialization, such as that proposed by HE et al. [He+15][18].

Backpropagation Error

As we have already discussed, the calculation of the gradient of the loss function $\nabla_\theta\, \mathcal{L}(\Omega,\theta)$ will be a fundamental element in the training of neural models. To do this, by virtue of what was discussed above (see, for example, Equation (2.18) on page 23), it will be sufficient to compute the gradient with respect to a single element of the dataset, that is, $\nabla_\theta\, \ell(x,y,\theta)$. This computation can be done efficiently using the well-known recursive algorithm *backpropagation*. This algorithm will depend on the

[15] Diederik P KINGMA and Jimmy BA. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint* (2014). eprint: `1412.6980` (cs.LG).

[16] Xavier GLOROT and Yoshua BENGIO. "Understanding the difficulty of training deep feedforward neural networks." In: *AISTATS*. ed. by Yee Whye TEH and D. Mike TITTERINGTON. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256. URL: `http://dblp.uni-trier.de/db/journals/jmlr/jmlrp9.html#GlorotB10`.

[17] For convolutional layers, $D_{\text{in}} = \texttt{filter\_size}^2 \times \texttt{input\_channels}$.

[18] K. HE, X. ZHANG, S. REN, and J. SUN. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: `10.1109/ICCV.2015.123`.

topology and type of neural model used, but it consists of the recursive application of the chain rule.[19]

Assuming we are facing a FNN, this backpropagation algorithm can be exemplified: let w^l_{ij} be an individual weight, which is involved in computing the output y^l_i according to Equation (2.3) on page 13, then the chain rule applied to $\ell(x, y, \theta)$ yields,

$$\frac{\partial \ell}{\partial w^l_{ij}} = \frac{\partial \ell}{\partial y^l_i} \frac{\partial y^l_i}{\partial w^l_{ij}}. \tag{2.25}$$

And, then, to compute the first part, the chain rule can be applied recursively again, in terms of $y^L, y^{L-1}, \dots, y^{l+1}$. A more rigorous approach and formalized version of the backpropagation algorithm is given by Bishop et al. [Bis+95][20] (see Section 4.8 for more details).

Computationally, this recursive process of gradient computation is performed using a computational graph and using techniques for automatic differentiation, rather than the techniques used to computationally compute derivatives repeatedly making use of the chain rule. PyTorch has a built-in differentiation engine called `torch.autograd` that supports automatic computation of gradient for any computational graph.

2.3.2 Regularization Techniques

One of the biggest problems facing DL and more generally predictive statistics is that of *overfitting*. Overfitting is creating a model of analysis from data that fits too closely the training dataset available, but not capable of making reliable predictions of future observations. In words of Neumann [Neu14][21],

> *With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.*
>
> —Neumann [Neu14]

What he is referring to here is that we should not be surprised at the ability—of a complex enough model—to fit a given dataset very well. Models with a large number of parameters are capable of adjusting to any given amount of data, even if it appears not to follow any pattern. But these settings will be meaningless, as they won't capture any genuine information on the structure of the data and, therefore, will not be able to generalize correctly. We will have an astonishingly good fit (even perfect in some cases) for currently available data, but with a huge generalization error (see Figure 2.10 on the facing page).

A well-known saying in the field of ML is that of "memorizing is *not* learning". It is important that we take into account the effect of overfitting at the time to train the

[19] In calculus, the chain rule is a formula to compute the derivative of a composite function. That is, if f and g are differentiable functions, then the chain rule expresses the derivative of their composite as $(f \circ g)' = (f' \circ g) \cdot g'$. This formula also applies for multi-variable functions.

[20] C.M. Bishop, P.N.C.C.M. Bishop, G. Hinton, and Oxford University Press. *Neural Networks for Pattern Recognition*. Advanced Texts in Econometrics. Clarendon Press, 1995. isbn: 9780198538646.

[21] John von Neumann. *Common Mistakes in using Statistics: Spotting and Avoiding Them*. `https://web.ma.utexas.edu/users/mks/statmistakes/ovefitting.html`. [Online; accessed 20 February 2021]. 2014.

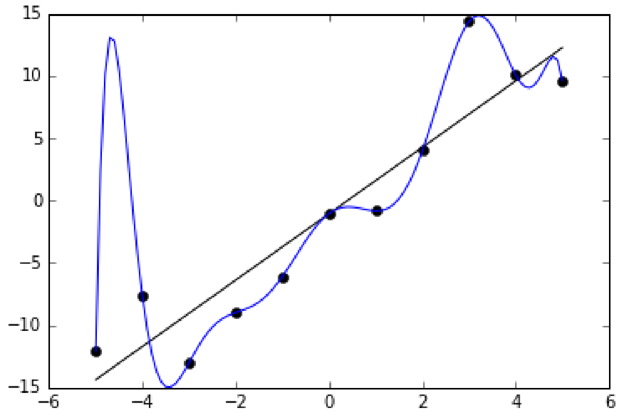


Figure 2.10. Representation of overfitting phenomenon. Noisy data (which is approximately linear) have been adjusted using a linear and a polynomial function. The polynomial function presents a perfect fit on existing data, but it is to be expected that it generalizes worse. If both functions were used to extrapolate out of the fitted data, the linear function will display better predictions. From *Overfitting — Wikipedia, The Free Encyclopedia*, by WIKIPEDIA CONTRIBUTORS [Wik21c].

models. To avoid this there are different techniques, among which highlight the L_2 regularization, the concept of early stopping and the data augmentation technique.

L_2 Regularization

This regularization technique, also known as *ridge regression* in statistics and *Tikhonov* regularization in mathematics, it is widely used in ML. It consists in adding a term to the loss function that takes into account the complexity of the model, i.e., minimize the following,

$$\hat{\mathcal{L}}(\Omega, \theta) = \mathcal{L}(\Omega, \theta) + \lambda \, \text{complexity}(\theta), \tag{2.26}$$

where λ is the regularization hyperparameter[22] and the complexity is measured using the L_2 norm, i.e.,

$$\text{complexity}(\theta) = \|\theta\|_2^2 = \sum_{w \in \theta} w^2. \tag{2.27}$$

Early Stopping

Probably the best known and most used method when it comes to training neural models. What you do is divide the dataset of *training* into two: the training part and the validation part (`train/val` split).[23]. For the training process only the part of `train` (i.e., for updating the parameters of the model). The part of `val` will be used to determine when we should stop the training process to avoid overfitting.

[22] Model *hyperparameters* are properties that govern the entire training process and must be set before the training starts (therefore, they are not trainable).

[23] Consequently, what to do with the full dataset is actually to divide it into three, the well-known `train/val/test` split.

In this way, as we iterate through the training process, we can compute the loss function both in the dataset of `train` and in that of `val` (see Figure 2.11). With the goal in mind of finding models that generalize well, we will stop training the neural network at the moment the error function in the validation dataset starts to increase. At the beginning of the training process the error function decreases in the two datasets of training and validation at the same time. At the time when the error function in the validation set begins to increase is when the overfitting problem, so we will stop the training there.

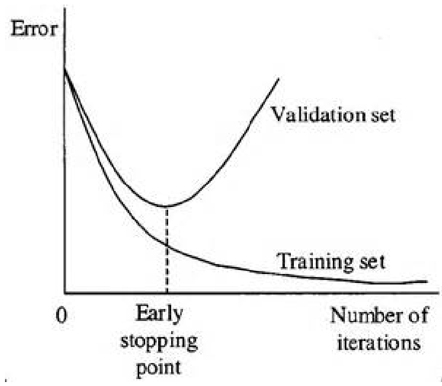


Figure 2.11. Early stopping regularization technique. The error in `val` and in `test` is shown and the separation of the graph is seen. From "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging", by Gençay et al. [GQ01].

This regularization method has been used in this work (see Chapter 4 on page 45) for model training.

Data Augmentation

The use of data augmentation as a regularization technique is the process of artificially generate new training samples from existing ones using random transformations. In image processing, more typical transformations are: color transformation, geometric translation, rotations, etc.

In this way we increase the cardinal of the dataset (which prevents the overfitting) and we also artificially force a more general dataset. This, consequently, will make the trained model better generalize and be invariant to the transformations performed in the dataset.

> **!** It is important not to overdo this technique, as it could lead to scenarios that are too synthetic and that, therefore, we generate models that do not work well in the real world. A qualitative evaluation of the data created can be useful to validate this technique.

Despite the possible problems that it may present, it is a technique commonly used for training various models within the scope of ML. This is mainly due to the fact that the data collection is a bottleneck in many jobs, due to its inherent difficulty and many times the associated economic cost. Data augmentation, however, is free, quick and easy to achieve by any sufficiently prepared team.

2.4 Testing

It is common to reserve a part of the dataset as a `test` dataset in order to evaluate the model. This division is part of the well-known `train`/`val`/`test` split. The `test` will not be used in any case to train the model, nor to validate it during training, nor to adjust the hyperparameters, nor for any other topic related to the training of a neural network. It will *only* be used to evaluate the model once it has already been trained, so that it can be compared with other models using some evaluation metric (typically quantitative). In this way, it will be possible to compare different models using *objective* criteria.

> ! Using the `test` dataset to train the model would lead to *false* evaluation metrics, since the model could be simply memorizing and not really learning (overfitting).

Good results in the `test` dataset confirm the model's ability to generalize and behave correctly in unknown scenarios (inputs never presented to the model).

We may hope that machines will eventually
compete with men in all purely intellectual fields.
—Alan Turing

Chapter 3

Referring Expression Comprehension

The task of Referring Expression Comprehension (REC) consists in, given a Referring Expression (RE)—is a linguistic phrase or human speech—and an image, generate a segmentation for the object which the phrase refers to (i.e., a binary mask). In this chapter we will specifically formulate the problem to be solved (see Section 3.1), we will analyze the existing datasets and the evaluation measures (see Section 3.2 on the following page) and finally, we will make an exhaustive study of state-of-the-art works in this area by reviewing the more recent literature (see Section 3.4 on page 41).

Regarding the nomenclature, some publications do not agree with the name to use. Several authors make use of the expression "Referring Expression *Segmentation*" instead of *"Comprehension"* to specify that the segmentation is being carried out and not just the generation of a bounding box. However, in this work we will use the term REC in the most general sense, possibly, encompassing both the models that generate the bounding box and those that generate the segmentation. It is clear that the step from segmentation to bounding box is trivial, while the opposite conversion has more complexity (but it can be done using neural models).

3.1 Problem Formulation

In the task of REC two different entries must be given, one of them related to language and the other to vision. Regarding the *vision* part, it can be an image or a video. In our case we will only deal with images.[1] It is also necessary a RE, that is a linguistic phrase that refers to an object. It can occur in two media: audio and text. In this thesis, both representations will be admitted. And, the output of this problem will be the generation of a binary segmentation mask with the referred object or a bounding box. In this thesis only the segmentation will be considered, but this

[1] The same model would also apply in the case of video if we worked frame by frame, but we will not offer a model that takes into account the temporal evolution of the frames.

is because is more general; generating the bounding box from the segmentation is trivial (which is not true in the other direction).

In order to understand it, multiple examples of this problem are shown in Figure 3.1 on the facing page. They have tried to show all the possibilities, from the simplest to the most complex.

- **Multiple objects.** In Figure 3.1a on the next page an example is shown with two people who differ from each other by a differentiating element (cap). In Figure 3.1b on the facing page the different objects are differentiated by their relative position (right). In Figure 3.1c on the next page they are differentiated by one quality (white suit).
- **Multiple categories.** As you can see in Figures 3.1g to 3.1i on the facing page, it can refer to both objects and people within the same image.
- **Specialized vocabulary.** In Figure 3.1e on the next page we refer to a specific type of animal (elephant) and in Figure 3.1f on the facing page the expression couch is used.
- **Secondary objects.** In Figure 3.1d on the next page we refer to a small secondary object in the image (car), which is part of the same category (car) as the main object (bus).

3.2 Training

There are two fundamental things in the process of training a model: a dataset and a loss function. For the specific task of this work (REC), we will present the most used datasets (see Section 3.2.1) and a list of loss functions (see Section 3.2.2 on page 35).

3.2.1 Datasets

There are different datasets created exclusively for the training and evaluation of neural models created to solve the problem discussed here. The first three datasets considered (RefCOCO, RefCOCO+ and RefCOCOg) take their images from the well-known dataset Common Objects in Context (COCO), created by Lin et al. [Lin+14][2], while the last one (CLEVR-Ref+) uses synthetic images.

The dataset COCO, as its name suggests, contains images of everyday life in everyday environments. Contains multi-object labeling, segmentation mask annotations, image captioning, key-point detection and panoptic segmentation annotations. They have a total of 81 categories, divided into 13 super-categories.

- **Super-categories.** They are as follows: person, vehicle, vehicle, outdoor, animal, accessory, sports, kitchen, food, furniture, electronic, appliance, indoor.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, et al. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755. arXiv: 1405.0312 [cs.CV].

(a) Man with cap

(b) Laptop on the right

(c) Army officer white suit

(d) Black car

(e) Small middle elephant

(f) Two seat couch

(g) Umbrella

(h) Parent holding umbrella

(i) Little girl pink coat

Figure 3.1. Examples of Referring Expression Comprehension. As you can see, we can refer to objects of the image with RE in natural language and segmentation occurs. Figures created by the author (all). View images in color to better appreciate segmentation.

- **Categories.** They are as follows: 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, donut, 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 drier, toothbrush.

Contains a total of 2 500 000 images.

RefCOCO, RefCOCO+ and RefCOCOg

These datasets were created from COCO by Kazemzadeh et al. [Kaz+14][3] using the game called ReferIt Game. In this two-player game, one of them wrote a RE based on an object in an image and the second player, given the image and RE, had to click on the correct location of the object that was being described. If the second user's click coincided in the correct region, each of the players received a point in the game and the roles were exchanged for the next image. In this way, a process was created in which RE were generated and validated for different objects in images in the same game.

The main difference between RefCOCO and RefCOCO+ is that in RefCOCO+ the *location* information was disallowed. In total, RefCOCO has 142 209 RE for a total of 50 000 objects in 19 994 images. RefCOCO+ has a similar number of RE objects and images. An example of each of these datasets is shown in Figure 3.2 on the facing page.

The difference between RefCOCOg and these two previous ones is that, it contains only elements *non-trivial*, i.e., there is at least one more object of the same class as the target object in the image. Regarding size, RefCOCOg contains 104 560 expressions, 54 822 objects and 26 711 images.

CLEVR-Ref+

The above datasets have been created expressly for REC and are made up of real-world images (which are highly complex). Furthermore, these datasets, whose images come from COCO, could be skewed. Therefore, Liu et al. [Liu+19b][4], created the CLEVR-Ref+ dataset with images and RE generated synthetically. Different situations are considered where objects are placed in the image with different variable options (such as colors, sizes, and spatial relationships).

[3] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L. Berg. "ReferIt Game: Referring to Objects in Photographs of Natural Scenes". In: *EMNLP*. Oct. 2014. url: `http://tamaraberg.com/referitgame/`.

[4] Runtao Liu, Chenxi Liu, Yutong Bai, and Alan L Yuille. "CLEVR-Ref+: Diagnosing Visual Reasoning with Referring Expressions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4185–4194. arXiv: `1901.00850 [cs.CV]`.

(a) RefCOCO dataset (b) RefCOCO+ dataset

Figure 3.2. Examples from RefCOCO and RefCOCO+ datasets. It can be seen that in RefCOCO+ the use of location information is not allowed, while in RefCOCO it is valid. Adapted from "ReferIt Game: Referring to Objects in Photographs of Natural Scenes", by Kazemzadeh et al. [Kaz+14].

This dataset, however, will not be used in the work that concerns this thesis, since here we are looking for models that work in the real world with non-fictional images and natural language.

3.2.2 Loss Functions

To train the models it is necessary to have a loss functions, that must be differentiable since in the optimization process we will need to use the partial derivatives of the loss function with respect to the different parameters to be trained.

In our case, we will always deal with two classes (the segmentation will be a binary mask).

Cross Entropy

One of the best known loss functions in image segmentation is that of Cross Entropy (CE). Next we will explain this loss function for a single pixel, but applied to a complete image it would consist of taking the arithmetic mean of each pixel. That is, we will define the function pixel-wise. For each pixel, we have two probability distributions.

1. **Prediction** can be $P(\hat{Y} = 0) = \hat{p}$ or $P(\hat{Y} = 1) = 1 - \hat{p}$.
2. The **ground truth** can either be $P(Y = 0) = p$ or $P(Y = 1) = 1 - p$. It will always be $p \in \{0, 1\}$.

The loss function is then defined as,

$$\mathrm{CE}(p, \hat{p}) = -(p \log \hat{p} + (1 - p) \log(1 - \hat{p})). \tag{3.1}$$

Taking into account that $p \in \{0, 1\}$, the loss function can be rewritten as follows,

$$\mathrm{CE}(p, \hat{p}) = \begin{cases} -\log(1-\hat{p}) & p = 0 \\ -\log \hat{p} & p = 1. \end{cases} \tag{3.2}$$

That is, if $p = 1$, the loss function will be 0 if and only if $\hat{p} = 1$ and it will be larger the more different p and $\hat{p}$ are. The penalty will grow exponentially until it becomes infinite for the value $\hat{p} = 0$. Case $p = 0$ is symmetric.

Various variations of this loss function will be discussed below that may be useful for training various neural models.

- **Weighted Cross Entropy (WCE).** It is a variant of CE in which the positive examples are weighted by a coefficient β. It is defined as follows,

  $$\mathrm{WCE}(p, \hat{p}) = -(\beta p \log \hat{p} + (1-p)\log(1-\hat{p})). \tag{3.3}$$

  Typically used when unbalanced classes appear. It is not too interesting for this case.

- **Balanced Cross Entropy (BCE).** It is similar to WCE with the only difference that a weight is also added to the negative examples. It is defined as follows,

  $$\mathrm{BCE}(p, \hat{p}) = -(\beta p \log \hat{p} + (1-\beta)(1-p)\log(1-\hat{p})). \tag{3.4}$$

- **Focal Loss (FL).** It is a variant of CE in which the most *complicated* elements of the dataset are affected even more. These are the ones with a value of $\hat{p}$ intermediate between 0 and 1. It is defined as follows,

  $$\mathrm{FL}(p, \hat{p}) = -(\alpha(1-\hat{p})^{\gamma} p \log \hat{p} + (1-\alpha)p^{\gamma}(1-p)\log(1-\hat{p})). \tag{3.5}$$

  When $\gamma = 0$ we obtain BCE.

- **Distance to the Nearest Cell (DNC).** This loss function, introduced by Ronneberger et al. [RFB15][5], forces the separation between contiguous objects. It is similar to BCE, but with an additional term of distance between objects,

  $$\mathrm{DNC}(p, \hat{p}) = -(w(p)\log \hat{p} + w(p)(1-p)\log(1-\hat{p})), \tag{3.6}$$

  where,

  $$w(p) = \beta + w_0 \cdot \exp\left(-\frac{(d_1(p) + d_2(p))^2}{2\sigma^2}\right). \tag{3.7}$$

  Here $d_1(p)$ denotes the distance to the edge of the nearest cell and $d_2(p)$ the distance to the edge of the second nearest cell. The rest are hyperparameters of the loss function.[6]

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention.* Springer. 2015, pp. 234–241. arXiv: `1505.04597 [cs.CV]`.

[6] The authors use $w_0 = 10$ and $\sigma \approx 5$ in their experiments (see Section 3 from [RFB15]).

Overlap Measures

Another type of measure arises with the use of the intersection and union of the predicted segmentation and the ground truth. This type of loss functions provide us with *global* information. The well-known Jaccard index or the Intersection over Union (IoU) coefficient,

$$J(A,B) = \frac{|A\cap B|}{|A\cup B|} = \frac{|A\cap B|}{|A|+|B|-|A\cap B|}, \tag{3.8}$$

is typically used to measure the accuracy of a model, but it cannot be used as a loss function as it is not a differentiable mapping. Yes, it will be used for the evaluation of the model in Section 5.1 on page 57.

- **Dice Loss (DL).** It is based on Dice Coefficient (DC), a coefficient similar to IoU, which is defined as follows,

  $$\mathrm{DC}(X,Y) = \frac{2|X\cap Y|}{|X|+|Y|}. \tag{3.9}$$

  This says coefficient can be defined as a loss function,

  $$\mathrm{DL}(p,\hat{p}) = 1 - \frac{2\sum p_{h,w}\hat{p}_{h,w}}{\sum p_{h,w} + \sum \hat{p}_{h,w}}, \tag{3.10}$$

  where $p_{h,w} \in \{0,1\}$, $0 \le \hat{p}_{h,w} \le 1$ and the sums are spread over the entire image at width w and height h.

- **Tversky Index (TI).** It is a generalization of DL. It is defined as follows,

  $$\mathrm{TI}(p,\hat{p}) = 1 - \frac{p\hat{p}}{p\hat{p} + \beta(1-p)\hat{p} + (1-\beta)p(1-\hat{p})}. \tag{3.11}$$

  Note that with the value $\beta = \frac{1}{2}$, we recover the previous function DL.

Generalized Intersection over Union Loss

See first Section 3.3.1 on the following page for a detailed explanation of the Jaccard index or IoU, which is a quantitative measure widely used as an evaluation technique.

> **!** The reason why IoU cannot be used directly as a loss function is that optimization is infeasible in the case of non-overlapping bounding boxes (since, in this case, IoU has no value and therefore no gradient).

Rezatofighi et al. [Rez+19][7] solve this problem by introducing a loss function based on IoU and which they call Generalized Intersection over Union (GIoU). This generalization guarantees the existence of a gradient in all cases and, therefore, makes it suitable for use in an optimization process.

[7] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, et al. "Generalized Intersection over Union: A metric and a loss for bounding box regression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* June 2019, pp. 658–666. arXiv: `1902.09630 [cs.CV]`.

Algorithm 1: Generalized Intersection over Union

input : Two arbitrary convex shapes: $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$
output: $GIoU$

1. For A and B, find the smallest enclosing convex object C, where $C \subseteq \mathbb{S} \in \mathbb{R}^n$
2. $IoU = \dfrac{|A \cap B|}{|A \cup B|}$
3. $GIoU = IoU - \dfrac{|C \backslash (A \cup B)|}{|C|}$

Figure 3.3. Generalized Intersection over Union (GIoU) general algorithm applied to arbitrary multi-dimensional sets. In the specific case of REC they will be bounding boxes in $\mathbb{R}^2$. From "Generalized Intersection over Union: A metric and a loss for bounding box regression", by Rezatofighi et al. [Rez+19].

This loss function is summarized in Figure 3.3. It is a generalization that preserves the relevant properties of IoU, but that corrects the problems related to its differentiability.

Combinations

Many more loss functions can be obtained by simple linear combination of the above. The combination,

$$\mathrm{CE}(p, \hat{p}) + \mathrm{DL}(p, \hat{p}), \tag{3.12}$$

is quite popular, since it combines local information (CE) with global information (DL).

3.3 Evaluation Techniques

In any area of Machine Learning (ML) it will be necessary to have evaluation techniques to be able to decide if the results obtained with the model are good enough and also to be able to make comparisons.

The most useful techniques are usually quantitative metrics, which will be discussed in Section 3.3.1. Furthermore, in the specific case of REC, due to the use of images, it will also be possible to carry out a visual or qualitative evaluation (see Section 3.3.2 on page 40).

3.3.1 Quantitative Measures

This corresponds to an evaluation of the model in a numerical way with metrics. The different evaluation measures typically used to address this problem are related to the computation of IoU or Jaccard index. This index is based on the concepts of intersection and union between the predicted segmentation (which is a *binary* mask) and the ground truth (diagrams with these concepts are shown in Figure 3.4 on the facing page).

(a) Union of sets A and B. **(b)** Intersection of sets A and B.

Figure 3.4. Graphic representation of the union and intersection of sets called A and B. From *Jaccard index — Wikipedia, The Free Encyclopedia*, by Wikipedia contributors [Wik21a] (both).

From here arises the well-known Jaccard index or coefficient IoU,

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \tag{3.13}$$

which is typically used to measure the accuracy of a model, but—as we have commented previously—it cannot be used as a loss function as it is not a differentiable application.

(a) Bounding boxes example **(b)** IoU visual equation

Figure 3.5. Explanation and example of the Jaccard index in the case of bounding boxes. From *Jaccard index — Wikipedia, The Free Encyclopedia*, by Wikipedia contributors [Wik21a] (both).

This index provides relevant information on how tight a bounding box is.[8] It is evident that the Jaccard index takes a value between 0 and 1, being 0 when there is no intersection between the bounding boxes and taking the value of 1 when the correspondence is exact.

[8] The case of bounding box is studied for simplicity, but the same concept applies in the case of pixel by pixel segmentation.

Mean and Overall IoU

Using IoU as a metric to evaluate a segmentation or bounding box can be done in two ways. The first is to average all the IoU values on the test dataset, as follows,

$$\text{Mean IoU} = \frac{1}{N}\sum_{i=0}^{N} \text{IoU}_i, \tag{3.14}$$

where N corresponds to the size of the test dataset and IoU_i is the IoU value corresponding to the i-th image. This metric, for obvious reasons, is called mean IoU (mIoU).

Another possibility for using the IoU as a metric is the *overall* IoU, defined as the division between the total intersection area and the total union area. This areas are accumulated by iterating throughout the dataset, i.e.,

$$\text{Overall IoU} = \frac{\sum_{i=0}^{N} I_i}{\sum_{i=0}^{n} U_i}, \tag{3.15}$$

where I_i and U_i correspond to the intersection and union (respectively) between the prediction and the ground truth for the i-th image in the test dataset.

One of the problems that *overall* IoU has is that it favors large regions like the ground or the sky.

Precision at Threshold

This metric is commonly used in segmentation task (e.g., is used in PASCAL Visual Object Classes challenge, by Everingham et al. [Eve+10][9]).

Here, for each sample from the test dataset, it will be judged as true/false positive by using the IoU index. A particular sample will be considered a correct detection iff the IoU between the prediction and the ground truth is greater than some predefined threshold. For example, Prec@0.5 is the percentage of samples where the predicted segmentation overlaps with the ground truth region by at least 50%.

Different thresholds can be used for evaluation, for instance computing Prec@0.5, Prec@0.7 and Prec@0.9. Of course, higher thresholds correspond to a harder metric and, thus, accuracy will decrease.

3.3.2 Qualitative Evaluation

In addition to making a quantitative evaluation, which provides us with numerical values for the different models, it is also interesting to carry out a qualitative evaluation.

In this specific case of REC this evaluation can be done visually, since it involves text and images. To perform this evaluation in the most general way it is important to test a very diverse range of both RE and images. To do this, RE can be used by involving different elements each time (and combining them): spatial information

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

within the image, name of the object, relative positioning, color characteristic of the object, position of the object, number of existing objects, relative size of the object, etc.

3.4 Related Work

The current state-of-the-art methods for REC can be divided into three main classes: joint embedding (see Section 3.4.1), modular models (see Section 3.4.2 on the following page) and graph convolution based models (see Section 3.4.3 on page 43).

3.4.1 Multimodal Embedding

Multimodal embedding methods are very typical in any of the multimodal learning tasks. What is sought in them is to find a multidimensional space where encodings of image and language can "coexist" in common. This idea is represented graphically in Figure 3.6. This multidimensional space will typically be $\mathbb{R}^n$, which is a normed space. One of the desirable characteristics would be that the encodings of images and language similar to each other were "close" in this space (in terms of norm).

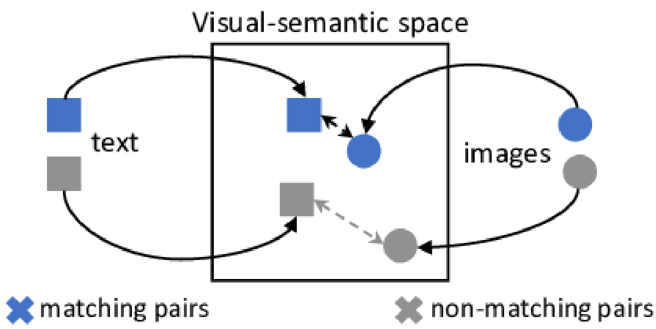


Figure 3.6. Multimodal embedding into visual-semantic space. As you can see, matching pairs are closer (in terms of norm) that non-matching pairs in the joint space. From "Towards Cycle-Consistent Models for Text and Image Retrieval", by Cornia et al. [Cor+18].

Therefore, here, to perform REC, the first thing we will do is encode the image and RE separately in the same vector space. For this, Convolutional Neural Network (CNN) are very useful to generate image representations (extracting the most relevant features) and for the coding of phrases Recurrent Neural Network (RNN) (with, for example, Long Short Term Memory (LSTM)) and transformers are used.

The first deep learning model for referring expression generation and comprehension is from Mao et al. [Mao+16][10], where they use a CNN model with which they extract the visual features and a network of type LSTM for *generating* the referring expression. It also gives a solution for the inverse problem of *comprehension*.

[10] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, et al. "Generation and Comprehension of Unambiguous Object Descriptions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 11–20. arXiv: `1511.02283 [cs.CV]`.

Within this type of model fits the one proposed by BELLVER et al. [Bel+20][11] where a neural network of type CNN is also used for the encoding of the image, but the transformer is used as a language encoder. Then to achieve multimodal embedding, the encoded linguistic phrase is converted into a 256-dimensional vector and multiplied element-wise with the visual features. This model will be studied in depth in Chapter 4 on page 45.

3.4.2 Modular Models

Modular models have been used successfully in many tasks both in the scope of Computer Vision (CV), and in Natural Language Processing (NLP). The technique used in these cases is to decompose REs into different components, in which it is sought to attack different reasoning.

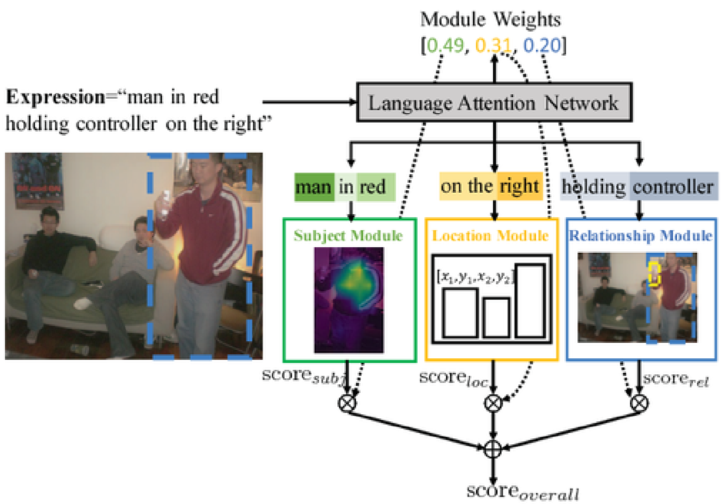


Figure 3.7. Modular Attention Network (MAttNet): given an expression, it is divided into three phrase embeddings, which are input to three visual modules that process the described visual region in different ways and compute individual matching scores. From "MAttNet: Modular Attention Network for Referring Expression Comprehension", by YU et al. [Yu+18].

An example of these modular models is the one presented by YU et al. [Yu+18][12], which is graphically represented in Figure 3.7. In this case there are three differentiated modules: the *subject* module, the *location* module and the *relationship* module. Each of them computes different scores, which are then used to calculate an overall score.

[11] Miriam BELLVER, Carles VENTURA, Carina SILBERER, Ioannis KAZAKOS, Jordi TORRES, et al. "RefVOS: A Closer Look at Referring Expressions for Video Object Segmentation". In: *CoRR* abs/2010.00263 (2020). arXiv: 2010.00263. URL: https://arxiv.org/abs/2010.00263.

[12] Licheng YU, Zhe LIN, Xiaohui SHEN, Jimei YANG, Xin LU, et al. "MAttNet: Modular Attention Network for Referring Expression Comprehension". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1307–1315. arXiv: 1801.08186 [cs.CV].

Starting from the base of MAttNet, LIU et al. [Liu+19c][13] propose Cross Modal Attention guided Erasing (CMAttErase), which is a training strategy for this type of task. It is based on the idea of erasing the part most used by the model from the linguistic or visual part, so that the model is forced to learn more complex structures.[14] Likewise, it modifies the initial model (MAttNet), considering the global image as one more characteristic.

3.4.3 Graph Generation

In the task that concerns us, the understanding by the model of RE is essential. These types of expressions contain different objects and relationships between them. In other words, it is common to refer to an object not only because of its intrinsic properties, but also because of its relationship with the objects that surround it. The mathematical tool that best represents this phenomenon is that of a graph: the nodes represent different objects and the different edges are the existing relationship between objects (see Figure 3.8).

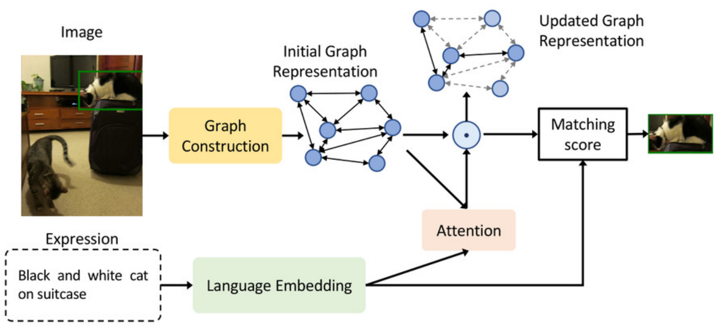


Figure 3.8. Summary representation of graph-based models. From the image, the graph representation is built, which is then updated with the expression embedding and computed a matching score between objects and expression. From "Referring Expression Comprehension: A Survey of Methods and Datasets", by QIAO et al. [QDW20].

The use of graphs in the task of REC has been used with success by various authors. Among them WANG et al. [Wan+19][15], he proposes Language Guided Graph Attention Network (LGRAN). This model consists of three differentiated modules: language-self attention module, language-guided graph attention module, and matching module. The first of these modules is responsible for decomposing

[13] Xihui LIU, Zihao WANG, Jing SHAO, Xiaogang WANG, and Hongsheng LI. "Improving referring expression grounding with Cross-modal Attention-guided Erasing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1950–1959. arXiv: `1903.00839 [cs.CV]`.

[14] It is partly similar to the strategy of *dropout* used in the training of fully connected neural networks, which is used to avoid dependency on specific neurons and thus prevent overfitting of the model.

[15] Peng WANG, Qi WU, Jiewei CAO, Chunhua SHEN, Lianli GAO, et al. "Neighbourhood Watch: Referring Expression Comprehension via Language-Guided Graph Attention Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1960–1968. arXiv: `1812.04794 [cs.CV]`.

the RE into three different parts (subject description, intra-class relationships, and inter-class relationships). The language-guided graph attention module is responsible for generating the graph representation of the image (the nodes it generates will be the candidate objects). Finally, the matching module is the one that computes the matching score between RE and object (for each of the candidate objects).

Other authors in exploiting the graphs in this context are YANG et al. [YLY19b][16], who created the model Dynamic Graph Attention Network (DGA), which allows multi-step reasoning. Initially, the model works the same as others with the generation of a graph from the image and with the mixing of an embedding of the expression in the graph. But from here on they use a module they call "analyzer" and that is capable of exploring the linguistic structure of RE and dividing it into a sequence of constituent expressions. In this way DGA is able to carry out a step-by-step reasoning process on these constituent expressions. Finally, as is common in these models (see Figure 3.8 on the preceding page), a matching score between objects and expression is computed.

YANG et al. [YLY19a][17] also create a graph-based model that they call Cross Modal Relationship Inference Network (CMRIN). This network consists of a Cross Modal Relationship Extractor (CMRE), which is in charge of obtaining the information for the construction of the graph with "cross-modal attention", and a Gated Graph Convolutional Network (GGCN) that uses the information from the previous graph and propagates the information (which is multi-modal) to be able to compute the matching score.

[16] Sibei YANG, Guanbin LI, and Yizhou YU. "Dynamic Graph Attention for Referring Expression Comprehension". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4644–4653. arXiv: `1909.08164 [cs.CV]`.

[17] Sibei YANG, Guanbin LI, and Yizhou YU. "Cross-Modal Relationship Inference for Grounding Referring Expressions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

All models are wrong, but some are useful.
—George Edward Pelham Box

Chapter 4

Models

Modeling consists of creating a mathematical model that represents a complex situation as closely as possible. In this work, two different models will be used: one of them to carry out the work of Referring Expression Comprehension (REC) starting from a Referring Expression (RE) in the form of text (see Section 4.1) and another model for speech recognition (see Section 4.2 on page 54), from so that you can also work with spoken natural language.

4.1 Referring Expression Comprehension

For the task of REC it will be necessary to find or create a neural model that solves it. To do this, we will start from a base architecture, i.e., a model to start with (see Section 4.1.1) and from there variations will be proposed—both in the model and in the way of training it—in Section 4.1.2 on page 48. That is, starting from the base model, an iterative process of improvement will be carried out.

4.1.1 Base Architecture

In Figure 4.1 on the next page a graphical representation of the model used as base architecture is shown. It has two differentiated parts with which the features are extracted from the visual part and from the language. These features are then combined to achieve a *multimodal* embedding and thus be able to generate the segmentation.

This model, created by Bellver et al. [Bel+20][1], will constitute our starting base architecture. Next, the image encoder (which is based on atrous convolutions), the language encoder (which uses transformers) and the multimodal embedding will be studied separately.

[1] Miriam Bellver, Carles Ventura, Carina Silberer, Ioannis Kazakos, Jordi Torres, et al. “RefVOS: A Closer Look at Referring Expressions for Video Object Segmentation”. In: *CoRR* abs/2010.00263 (2020). arXiv: `2010.00263`. URL: `https://arxiv.org/abs/2010.00263`.

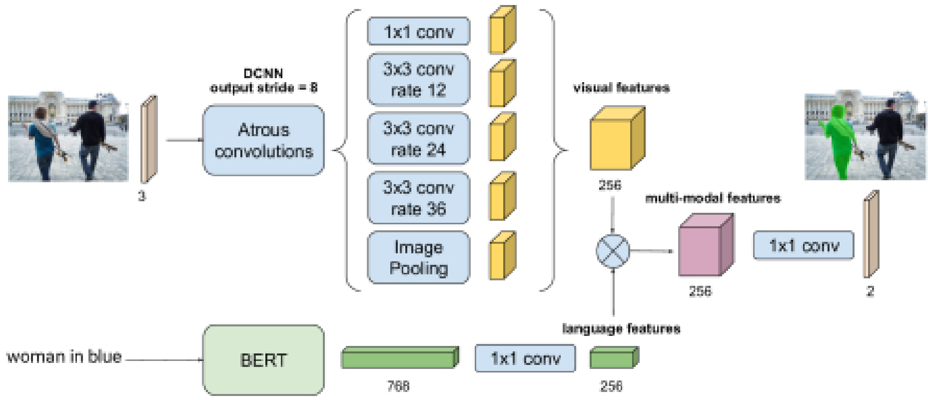


Figure 4.1. Referring Expressions for Video Object Segmentation (RefVOS) model architecture. It is possible to observe the differentiated models of vision and language that are then combined to obtain the multimodal characteristics. From "RefVOS: A Closer Look at Referring Expressions for Video Object Segmentation", by Bellver et al. [Bel+20].

Image Encoder

To extract the features of the images a state-of-the-art model called DeepLab is used, which is a neural network created by Chen et al. [Che+17][2] and based on atrous convolutions (see Figure 4.2). It is a Convolutional Neural Network (CNN) used for semantic segmentation.

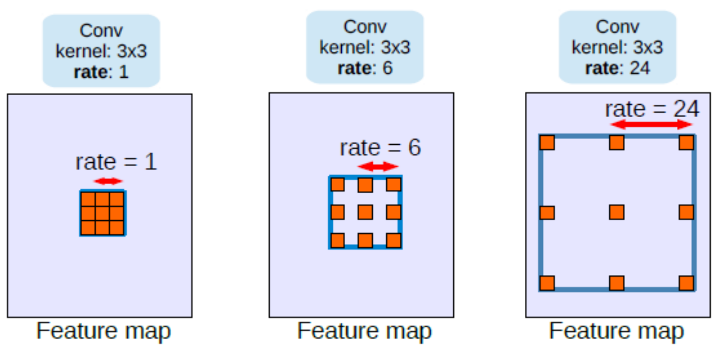


Figure 4.2. Atrous convolutions examples with filter size 3×3. The `rate` parameter controls the model's field-of-view. Standard convolution operation corresponds to an atrous convolution with a rate of 1. From "Rethinking Atrous Convolution for Semantic Image Segmentation", by Chen et al. [Che+17].

One of the advantages of this model compared to standard convolutional neural models is that it adapts very well to objects at different scales, without the need for

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv* (2017). arXiv: `1706.05587 [cs.CV]`.

pooling operations. Thus, the creators of this model define atrous convolutions (also known as dilated convolutions).

> *Atrous convolution allows us to extract denser feature maps by removing the downsampling operations from the last few layers and upsampling the corresponding filter kernels, equivalent to inserting holes ("trous" in French) between filter weights.*
>
> —CHEN et al. [Che+17]

In the model used, the well-known ResNet101 network (created by HE et al. [He+16][3]) and a `output_stride`[4] of 8. Likewise, $(12, 24, 36)$ will be used as `rates` of the convolutions in Atrous Spatial Pyramid Pooling (ASPP). These pyramids are part of the DeepLab model and consist of performing atrous convolutions in parallel (with different rates). In this way, by using different rates, it is possible to capture information from different scales at the same time.

Language Encoder

In the case of the language encoder, different possibilities could be considered, including using a Recurrent Neural Network (RNN) or mainly using a transformer. In this base architecture presented, RefVOS achieves more promising results by making use of transformers. Specifically, a transformer created by DEVLIN et al. [Dev+19][5] and called Bidirectional Encoder Representations from Transformers (BERT) is used.

BERT is a multi-layer bidirectional Transformer encoder (see Section 2.2.4 on page 20) that removes the unidirectional constraint present in previous models related to language representation. It uses Masked Language Model (MLM), i.e., randomly masks some tokens from the input and tries to predict the original token of masked word (just relying on its context). This allows the model to learn from both left and right context.[6]

To integrate BERT within the model, it is necessary to convert each of the REs to tokens and add two special tokens: `[CLS]` and `[SEP]` at the beginning and end of the sentence respectively. This model will then produce embeddings of dimension 768 for each of the input tokens. The final hidden vector corresponding to the first input token (`[CLS]`) as the aggregate representation of the RE (view section 4.1 from [Dev+19]) will be taken.

[3] Kaiming HE, Xiangyu ZHANG, Shaoqing REN, and Jian SUN. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778. arXiv: `1512.03385 [cs.CV]`.

[4] The `output_stride` is the ratio of input images partial resolution to final output resolution will be used as backbone. Setting this ratio to smaller values allow the model to extract denser feature responses (view section 3.1 from [Che+17]).

[5] Jacob DEVLIN, Ming-Wei CHANG, Kenton LEE, and Kristina TOUTANOVA. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Vol. 1. Minneapolis, Minnesota: Association for Computational Linguistics, May 2019, pp. 4171–4186.

[6] The model BERT also uses the task of Next Sentence Prediction (NSP) as an objective training function (see Task #2 in section 3.1 from [Dev+19]).

Multimodal Embedding

Once we have the encoded RE and the map of visual features from the convolutional network, it is necessary to obtain a multimodal embedding, which combines the information from both encoders. The output from the visual encoder is a tensor of depth 256 and the output from the language encoder is a 768-dimensional vector (see Figure 4.1 on page 46).

To combine these two outputs, the encoded RE of the vector of dimension 768 is converted to one of dimension 256 (which coincides with the depth of the visual features). These two tensors are then multiplied element-wise to obtain the multimodal embedding. Finally, a convolutional layer is used to pass a last tensor with two classes, which separate the *background* from the *object* that is being referred.

4.1.2 Model Iterations

Now, in this section, what we will try to do is understand the operation of the base architecture explained and proceed to carry out an iterative process of improvement of said model. For this we will attack the fundamental constituent parts of any neural model: change the architecture or change the way of training. As we know, regarding the architecture, in this case, we have three different parts (the image encoder, the language encoder and the multimodal embedding). And, regarding the training of the model, different parts can also be distinguished: loss function, criteria to stop training, optimization technique, use of pre-trained parameters, etc.

Loss Functions

Originally the function used for training is that of Cross Entropy (CE), however, for the segmentation task (specifically for binary segmentation as is this case) there are many more (see Section 3.2.2 on page 35). Within this entire list there are many of them that are variations precisely of CE. Training a model with these variations is of little significance in terms of results. Here, comment that no significant results are found, since, as it is a pre-trained model and the loss functions are quite similar, there is no progress in training, reaching an area where the gradients are practically zero.

Other loss functions that could be more interesting are those based on overlap measures. Among them we highlight that of Dice Loss (DL), which has been used to train the model based on pre-trained parameters. This loss function, which has already been defined in Section 3.2.2 on page 35, can be implemented in PyTorch as follows.

```python
def dice_loss(inputs, targets, smooth=1):
    """Dice Loss function implementation.

    Inputs and targets must be presented. Smooth is auxiliary value."""

    intersection = (inputs * targets).sum()

    num = 2.*intersection + smooth
    den = inputs.sum() + targets.sum() + smooth
```

```
    dice = num/den

    return 1 - dice
```

Using this loss function the evaluation of the training of the model is collected in Figure 4.3 on the next page. The training process is decided to stop at the moment when the loss function is lowest in the split `val`. It is important to observe the magnitudes on the vertical axis of the graph, since the variations are insignificant. It might seem at first glance that the evolution of the loss function is quite satisfactory due to the shape of the graph, but it must be taken into account that the scale of the vertical axis is extremely small, so the variations of the model parameters are really insignificant.

In order to better understand this evolution of the model, the overall Intersection over Union (IoU) on this same training process has also been plotted (see Figure 4.4 on the following page). In it we can see a very important presence of noise, variations really without any direction and without presenting a clear trend. It is also important to highlight in this case that the scale of the vertical axis is quite small: this same graph on a vertical axis in the range $(0, 1)$ would be practically flat.

Taking into account that the best epoch corresponds to the number 21 (see Figure 4.3 on the next page), we would obtain a value of overall IoU in this lower than that obtained by the initial model (see Table 4.1 on page 52). Yes, it is true that we could decide to take the values of the parameters at another time taking into account the peak of overall IoU at time number 16. Now, this graph actually presents very small fluctuations that are due to simple noise produced by the slight variation of the parameters when training and does not represent a significant improvement in the model.

In addition to this loss function, another one studied in Section 3.2.2 on page 35 has been tested, such as the one related to Tversky Index (TI). At first, certain improvements were expected in the overall IoU metric as it is a new loss function to optimize that could improve the model. The training process with this new loss function has been analogous to the one carried out previously, i.e., the model with pre-trained parameters is taken, the loss function is changed and the training process is “restarted” again. To do this, the loss function has been implemented in Pytorch as follows: shown as a file below.

```python
def tversky_loss(inputs, targets, smooth=1, alpha=0.5, beta=0.5):
    """Tversky loss function implementation"""

    # Flatten label and prediction tensors.
    inputs = inputs.view(-1)
    targets = targets.view(-1)

    # True positives, false positives and false negatives.
    TP = (inputs * targets).sum()
    FP = ((1-targets) * inputs).sum()
    FN = (targets * (1-inputs)).sum()

    tversky = (TP + smooth) / (TP + alpha*FP + beta*FN + smooth)

    return 1 - tversky
```

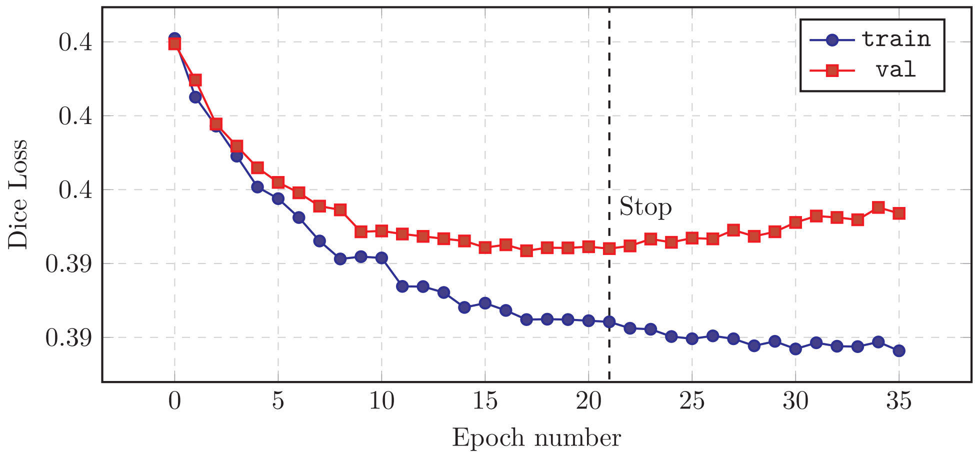


Figure 4.3. Training graph with Dice Loss. The evolution of the loss function for the different epochs for the `train/val` splits is shown. We are left with the epoch that presents a lower value of the loss function in the split `val`. Figure created by the author.

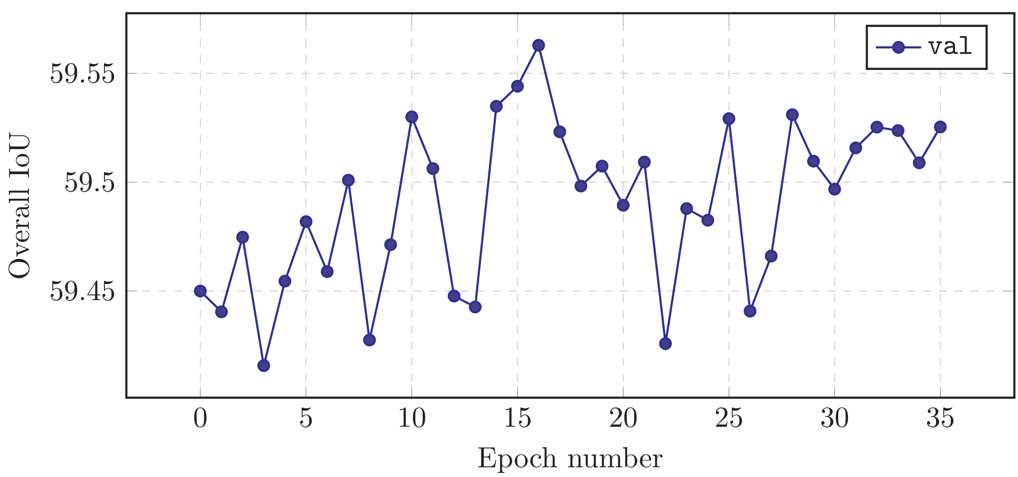


Figure 4.4. Overall IoU graph with Dice Loss. In this case, under the same evolution of the model optimizing the DL function, it is shown how the overall IoU evolves. Figure created by the author.

Unfortunately, again the results, despite the fact that the optimization process has reduced the value of the loss function during the iteration in epochs, it has not been possible to substantially improve the performance of the model in the overall IoU. More specifically, the improvement was less than 1%[7], which is not really an improvement with enough weight to put it in value. One of the problems encountered during this training process has been similar to that found in the previous case. The graph of the loss function—despite decreasing—has done so in a small way (that is, on a microscale so to speak). In this way, a graph similar to a successful training process has been achieved (it has stopped at the minimum achieved with the loss function in the split of `val`), but the real variation of the loss function has been really small. In addition, in a similar way, the precision or accuracy function presents too much noise between epochs: many variations up and down in the value, but without significant improvements (or worsening), which is what was really being sought.

Also comment that different loss functions can always be used to train the model. Now, typically the most used among them is CE, which is one of the ones that usually works best. Normally, the loss function change is not performed unless there is some compelling reason to do it this way. We have tried to improve the training process of the model by changing these functions a bit innocently, and it has not worked very well. This is probably due to the fact that the different loss functions lead to similar points, since they "seek" the same thing in the model: to improve segmentation.

Multimodal Embedding

Regarding multimodal embedding, there are different possibilities that can be carried out to obtain joint information both in terms of vision and language. Among them, those studied by the model RefVOS are those of addiction, multiplication and concatenation. That is, we can join the visual features and the language features with an element-wise operation. These different strategies are shown in Table 4.1 on the following page evaluated using the overall metric IoU in the RefCOCO dataset in the splits of `val/testA/testB`. As we can see, the fusion strategy that obtains a superior performance is that of *multiplication*, so it will be the one used in the future. In the original publication of the RefVOS paper, these comparative values did not appear, so they have been calculated to confirm the theory present in their work.

These three multimodal feature fusion strategies have in common that they are presented in an "arbitrary" way, so it was studied as an improvement that this multimodal fusion was learned by the model and not imposed externally. That is, it was tried that the fusion of features was learned by the model using data. To do this, using the notation V for the visual features tensor and L for the language features tensor, we have that their dimensions are $w \times h \times d$ and d respectively (w and h represent the width and the height of the visual features respectively). Then, the idea of following an approach similar to the one proposed by FAGHRI et al. [Fag+18][8]

[7] The improvements in the performance of the model have been specifically 0.47%, which for the `val` split of the RefCOCO dataset consists of a negligible increase in the overall IoU. It is also true that due to the noise present in this graph, really some other time had a higher performance.

[8] Fartash FAGHRI, David J FLEET, Jamie Ryan KIROS, and Sanja FIDLER. "VSE++: Improving Visual-Semantic Embeddings with Hard Negatives". In: *Proceedings of the British Machine Vision Conference (BMVC)*. July 2018. URL: `https://github.com/fartashf/vsepp`.

Table 4.1. Fusion strategies performance in RefCOCO dataset. The overall IoU for each fusion strategy for visual and language features is shown for the `val/testA/testB` splits in the RefCOCO dataset. Table created by the author.

	RefCOCO		
Strategy	`val`	`testA`	`testB`
Addition	56.60	60.87	51.29
Multiplication	**59.45**	**63.19**	**54.17**
Concatenation	55.12	58.88	49.59
Projection	Infeasible	Infeasible	Infeasible
Projection v2	21.08	-	-

arises, where linear porjections are defined from the features to an embedding space. To do this, it is necessary to reshape the visual features tensor and think of it as a vector $V \in \mathbb{R}^{w\times h\times d}$, (we will define to simplify the notation $D := w \times h \times d$, so we will write $V \in \mathbb{R}^D$). And, we will also use the vector of language features $L \in \mathbb{R}^d$. In this way, it is now possible to define applications to map features to a vector space of common dimension J. That is, the application ϕ is defined,

$$\begin{aligned} \phi\colon \mathbb{R}^D \times \mathbb{R}^{D\times J} &\longrightarrow \mathbb{R}^J \\ (V, W_v) &\longmapsto \phi(V, W_v) := W_v V, \end{aligned} \tag{4.1}$$

which maps the visual features V to the joint space $\mathbb{R}^J$ via the linear projection defined by the matrix of visual parameters W_v. In the same way, the application ψ is defined,

$$\begin{aligned} \psi\colon \mathbb{R}^d \times \mathbb{R}^{J\times d} &\longrightarrow \mathbb{R}^J \\ (L, W_l) &\longmapsto \psi(L, W_l) := W_l L, \end{aligned} \tag{4.2}$$

which maps the language features L to the joint space $\mathbb{R}^J$ via the linear projection defined by the language parameter matrix W_l.

Some decisions had to be made, including the decision of the size of the joint space D. Taking into account that the vector of language features had dimension d and that it would not be useful to propose a reduction in dimensionality or increase it (since the model already has enough complexity and free trainable parameters), it was decided to fix that it would not be used of parameters and $\psi = \mathrm{Id}$. Therefore, it only remained to add the function ϕ, which was completely defined by the matrix of visual weights W_v. Now, this initial idea of projection that seemed very useful, was found to be *infeasible* due to the enormous size of this matrix and the impossibility of training this huge number of parameters due to limited computational resources. We must take into account that $W_v \in \mathbb{R}^{D\times J}$, where $D = w \times h \times d$, and we have chosen $J = d = 256$ for $\psi = \mathrm{Id}$. In other words, the number of parameters in W_v is on the order of billions, which makes it computationally infeasible.

Once this problem has been detected, another similar approach is proposed, but drastically reducing the number of parameters. To do this, it is proposed to reuse parameters in the depth of the image characteristics (and continue using $\psi = \mathrm{Id}$). That is, use an array for each of the slices in the depth of the visual features. This is for each slice V^i in the depth of the visual features tensor, a weight matrix W_v is used (the same for all slices), so that the corresponding embedding for slice i is defined as follows $\tilde{V}^i = W_v V^i$. That is, in this case each of the visual features matrices is not extended as a vector, but rather is multiplied. To keep the original dimensions, use $W_v \in \mathbb{R}^{w \times h}$. In this way, it was possible to send the visual features tensor V to another modified visual features tensor $\hat{V}$ through a linear projection with trainable parameters. Later, the multiplication of features was used again to now achieve the fusion with the language (multiplication is used because it is what had been shown as more efficient before). Now, what was obtained from these variations to the model? Pretty bad results. Specifically, in the split of `val` in the RefCOCO dataset an overall IoU value of 21.08 was obtained, which is significantly lower than the values obtained by other techniques, so it was not decided to use this technique. After a reasoning of the method of the method used, the causes that cause problems in this case are:

- **Loss of spatial information.** The first and most important thing is to highlight that this technique described here causes the loss of spatial information that comes from the beginning from the image (and this is preserved by the convolution operations). That is, when performing the multiplication operation between matrices we are at the end combining "pixels" from different parts of the image without too much success.

- **Meaningless transformation to visual information.** Another problem that arises in the application of this technique is that it does not contribute significantly to the multimodal fusion between characteristics. Rather, it is an addition to the convolutional network used, which is not useful, since the model used is a well-known state-of-the-art model. In other words, we are adding one more layer without much sense to an existing model whose topology has already been precisely selected by its creators.

- **Adding unnecessary non-pretrained parameters.** Besides that, we have the problem that a significant number of parameters are being added (trainable, yes), but they are not pre-trained. That is, we are giving the model the ability to more easily overfit with these new parameters that have been added.

Training Process

Another possibility to modify is the model training process. This consists of modifying the optimization algorithm so that the development of the model can be improved. In Section 2.3 on page 21 different possibilities have already been discussed. In the original case of RefVOS, the optimizer Stochastic Gradient Descent (SGD) is used with Nesterov momentum 0.9 and `weight_decay` of 1×10^{-6}, this corresponds to the regularization L_2 (see Section 2.3.2 on page 26). Another possibility to consider would be Adam's optimizer, for example with the hyperparameters typical of $\beta_1 = 0.9$, $\beta_2 = 0.999$. However, it has not been recommended to change this optimization process, since it is not usual for it to report significant improvements.

4.2 Speech Recognition

Speech recognition, also known as automated speech recognition and Speech to Text (STT) is a field of Computer Science (CS) that deals with recognition of spoken language into text. For us it will be useful because it will allow us to segment objects in images using the voice, that is, we will be able to solve the problem of REC using spoken language.

For this task, we will use a pre-trained neural model to convert from STT. The model, created by Veysov [Vey20][9], is called Silero (see Figure 4.5 on the next page), and it allows converting from mono audio to text in different languages: English, German, Spanish and Ukrainian.

[9] Alexander Veysov. "Toward's an ImageNet Moment for Speech-to-Text". In: *The Gradient* (2020).

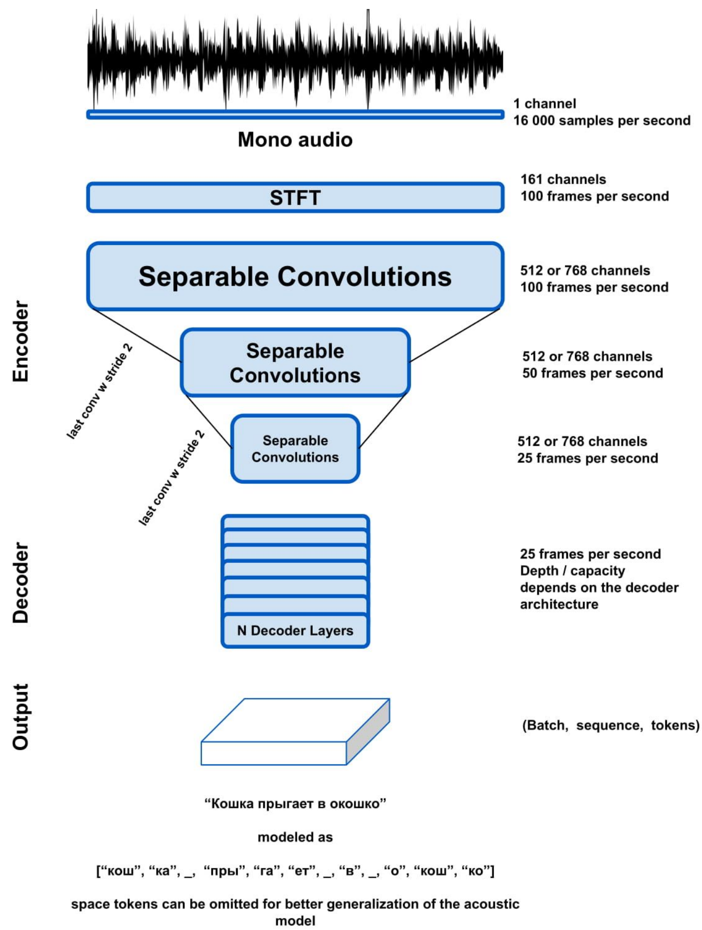


Figure 4.5. Silero Speech to Text (STT) model architecture. View from top to bottom: input is a mono audio file with speech and the output is the text representing the input. From "Toward's an ImageNet Moment for Speech-to-Text", by VEYSOV [Vey20].

However beautiful the strategy,
you should occasionally look at the results.
—Winston Churchill

Chapter 5

Results and Comparison

The results obtained with the model described in Chapter 4 on page 45 will now be studied and compared with other state-of-the-art models. Here the evolution of the model in its different iterations will not be shown, but only the last version selected will be considered. The evaluation will be carried out both quantitatively (see Section 5.1) and qualitatively (see Section 5.2 on page 62).

5.1 Quantitative Evaluation

Regarding the quantitative evaluation of the model (as already discussed in Section 3.3.1 on page 38) there are different metrics to use. Among them, three stand out: the mean and overall Intersection over Union (IoU), and precision at threshold. In the case of the model created in this work, we can evaluate it with any metric that we consider appropriate and in any dataset, since we have its implementation. Now, what is really interesting is being able to compare it with other state-of-the-art models that currently exist. The literature consulted in this work typically uses two fundamental metrics: overall IoU and precision at 0.5. The Prec@0.5 is used as a measure of accuracy, i.e., the number of percentage of samples where the predicted segmentation overlaps with the ground truth region by at least 50% is computed.

In this section we will show comparative tables of the quantitative evaluation of the model of this work and of other current models. For this, a study of the overall IoU (see Section 5.1.1) and a study of the accuracy or Prec@0.5 (see Section 5.1.2 on page 60) will be carried out. Unfortunately, it was not possible to present the same number of models in both two sections, due to the absence of these evaluation metrics in the original publications of the models.

5.1.1 Overall Intersection over Union

Regarding the overall IoU, data have been collected from multiple models, which are shown in a summarized way in Table 5.1 on page 59. The evaluation has been carried out on the RefCOCO and RefCOCO+ datasets with the splits `val/testA/testB`. As previously mentioned, the RefCOCO+ dataset presents Referring Expressions

(REs) of greater complexity, therefore it presents lower overall IoU values than in the RefCOCO dataset for all models.

Several of the evaluated models have been previously described in Section 3.4 on page 41. It should be noted that the differences between the overall metric IoU are not too large comparatively by model. You can see an approximate range of 50–60 for the RefCOCO dataset and 40–50 for the RefCOCO+ dataset (which has more complex expressions). The model presented in this work outperforms some of the models in the RefCOCO dataset and remains close to the state of the art. However, in the RefCOCO+ dataset it presents less promising results.

The absolute winner regarding this metric is the model created by Huang et al. [Hua+20][1] and called Cross-Modal Progressive Comprehension (CMPC). It presents the highest values of overall IoU in all the categories. This complex reasoning method is based on a multi-step structure. They explain their method in a summarized way.

> *The CMPC module first employs entity and attribute words to perceive all the related entities that might be considered by the expression. Then, the relational words are adopted to highlight the correct entity as well as suppress other irrelevant ones by multimodal graph reasoning. In addition, we further leverage a module to integrate the reasoned multimodal features from different levels with the guidance of textual information. In this way, features from multilevels could communicate with each other and be refined based on the textual context.*
>
> —Berners-Lee [Ber21]

The model CMPC, despite being superior in terms of this metric, also presents a considerably higher complexity than the model presented in this work. Similarly, the model Bi-directional Relationship Inferring Network (BRINet), created by Hu et al. [Hu+20][2], is superior in terms of performance, but in exchange for presenting greater complexity. Precisely, the model presented in this thesis consists of a simpler and fully end-to-end network, which presents results that are competitive with the current state of the art.

Also, it should be noted that the proposed model outperforms the Maximum Mutual Information (MMI) model created by Mao et al. [Mao+16][3]. This specific comparison is interesting because this model is also based on a joint embedding of language and image.

[1] Shaofei Huang, Tianrui Hui, Si Liu, Guanbin Li, Yunchao Wei, et al. “Referring Image Segmentation via Cross-Modal Progressive Comprehension”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 10488–10497. arXiv: `2010.00514 [cs.CV]`.

[2] Zhiwei Hu, Guang Feng, Jiayu Sun, Lihe Zhang, and Huchuan Lu. “Bi-Directional Relationship Inferring Network for Referring Image Segmentation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2020, pp. 4423–4432. doi: `10.1109/CVPR42600.2020.00448`.

[3] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, et al. “Generation and Comprehension of Unambiguous Object Descriptions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 11–20. arXiv: `1511.02283 [cs.CV]`.

Table 5.1. Overall Intersection over Union model comparison. For each of the models the overall IoU is shown for the splits `val/testA/testB` in the datasets RefCOCO and RefCOCO+. The state of the art in each category is shown in bold. Full names for model acronyms can be found in section *Model Acronyms* on page xix. Table created by the author using data from second column references.

		RefCOCO			**RefCOCO+**		
Method	**Paper**	`val`	`testA`	`testB`	`val`	`testA`	`testB`
ASGN	[Qiu+20]	50.46	51.20	49.27	38.41	39.79	35.97
BRINet	[Hu+20]	61.35	63.37	59.57	48.57	52.87	42.13
CAC	[Che+19b]	58.90	61.77	53.81	-	-	-
CMPC	[Hua+20]	**61.36**	**64.53**	**59.64**	**49.56**	**53.44**	**43.23**
CMSA	[Ye+21]	58.32	60.61	55.09	43.76	47.60	37.89
DMN	[Mar+18]	49.78	54.83	45.13	38.88	44.22	32.29
MAttNet	[Yu+18]	56.51	62.37	51.70	46.67	52.39	40.08
RefVOS	[Bel+20]	59.45	63.19	54.17	44.71	49.73	36.17
RMI	[Liu+17]	45.18	45.69	45.57	29.86	30.48	29.50
RRN	[Li+18]	55.33	57.26	53.95	39.75	42.15	36.11
STEP	[Che+19a]	60.04	63.46	58.97	48.18	52.33	40.41

Note. Models arranged in alphabetical order.

5.1.2 Accuracy or Precision at 0.5

Regarding the accuracy or Prec@0.5, a comparative study has also been made, which is shown in Table 5.2 on the facing page. It should be remembered that Prec@0.5 consists of computing the number of percentage of samples where the predicted segmentation overlaps with the ground truth region by at least 50%. Unfortunately the comparison of models in this section is not easy due to the significant lack of data for some models. For example, the state-of-the-art model in the previous section (CMPC) only presents accuracy data for the split of `val` in RefCOCO. Likewise, the state-of-the-art model in RefCOCO+ (shown in bold) does not present data for the RefCOCO dataset, which makes comparison considerably difficult.

Despite this, as can be seen, in the RefCOCO dataset the model that presents a superior performance is that of Cross Modal Attention guided Erasing (CMAttErase), created by LIU et al. [Liu+19c][4]. This model is mainly based on a training strategy based on the idea of eliminating the parts most used by the model from the linguistic or visual part, so that it is forced to learn more complex structures. It must be taken into account that despite this model being the one that obtains the highest accuracy values, it could possibly be outperformed by the model ViLBERT (of which, unfortunately, no evaluation data is available for this dataset).

In the RefCOCO+ dataset, the model Vision-and-Language BERT (ViLBERT), created by LU et al. [Lu+19][5], is proclaimed as the winner and, therefore, state of the art. Broadly speaking, it consists of reusing the well-known and popular architecture of Bidirectional Encoder Representations from Transformers (BERT) to a multimodal model with visual and textual inputs that interact with each other using co-attentional transformer layers. It is a very interesting approach, not only for this specific task, but also for the field of multimodal learning in general and this is what its authors express.

> *Our work represents a shift away from learning groundings between vision and language only as part of task training and towards treating visual grounding as a pretrainable and transferable capability.*
>
> —LU et al. [Lu+19]

As we can see, the model presented in this work is not the state of the art, but it presents quite reasonable precision values. As with the rest of the models, the precision decreases in the RefCOCO+ dataset due to the added complexity in the REs, as previously mentioned. Yes it is true, that there is still a lot of room for improvement in the field of precision, but the results are quite promising.

The standardized accuracy metric is Precission at 0.5 (Prec@0.5), now this could be done with different thresholds (0.6, 0.7, 0.8, etc.) and the accuracy should decrease

[4] Xihui LIU, Zihao WANG, Jing SHAO, Xiaogang WANG, and Hongsheng LI. "Improving referring expression grounding with Cross-modal Attention-guided Erasing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1950–1959. arXiv: `1903.00839 [cs.CV]`.

[5] Jiasen LU, Dhruv BATRA, Devi PARIKH, and Stefan LEE. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *arXiv preprint* (2019). eprint: `1908.02265` (cs.CV).

Table 5.2. Accuracy or Prec@0.5 model comparison. For each of the models the accuracy percentage or Prec@0.5 is shown for the splits `val/testA/testB` in the datasets RefCOCO and RefCOCO+. The state of the art in each category is shown in bold. Full names for model acronyms can be found in section *Model Acronyms* on page xix. Table created by the author using data from second column references.

		RefCOCO			**RefCOCO+**		
Method	**Paper**	`val`	`testA`	`testB`	`val`	`testA`	`testB`
BRINet	[Hu+20]	71.83	75.09	68.38	-	-	-
CAC	[Che+19b]	77.08	80.34	70.62	-	-	-
CMAttErase	[Liu+19c]	**78.35**	**83.14**	**71.32**	68.09	73.65	58.03
CMPC	[Hua+20]	71.27	-	-	-	-	-
CMSA	[Ye+21]	69.24	73.87	64.55	45.48	51.41	37.57
FAOA	[Yan+19]	71.15	74.88	66.32	56.88	61.89	49.46
LGRAN	[Wan+19]	-	76.6	66.4	-	64.00	53.40
MAttNet	[Yu+18]	76.65	81.14	69.99	65.33	71.62	56.02
MMI	[Mao+16]	-	64.90	54.51	-	54.03	42.81
NMTree	[Liu+19a]	74.71	79.71	68.93	65.06	70.24	56.15
RefVOS	[Bel+20]	67.34	70.47	65.02	57.28	60.31	46.37
RMI	[Liu+17]	42.99	42.99	44.99	20.52	21.22	20.78
RRN	[Li+18]	61.66	64.13	59.35	37.32	40.80	32.42
STEP	[Che+19a]	70.15	-	-	-	-	-
ViLBERT	[Lu+19]	-	-	-	**72.34**	**78.52**	**62.61**

Note. Models arranged in alphabetical order.

as this value increases, since each instead we look for a more perfect segmentation to consider it as a positive sample. A study for a different threshold will not be presented here for the simple reason of lack of data for the models studied: few or none of the publications do a study and present their precision results for different threshold values.

5.2 Qualitative Evaluation

This work can also be easily evaluated qualitatively, since the result of the segmentation can be seen graphically superimposed on the input image. Overall, the model proposed in this work considerably well at this task with fairly consistent and accurate results. The reader can go to the website[6] of this project to test for himself the operation of the model. Here we will show several examples where different images have been used and REs very varied. More specifically, successful results will be shown in Section 5.2.1 and also, a study of examples will be made in which the model fails—or does not achieve a sufficiently precise segmentation—in Section 5.2.2 on page 64.

5.2.1 Study of Successful Samples

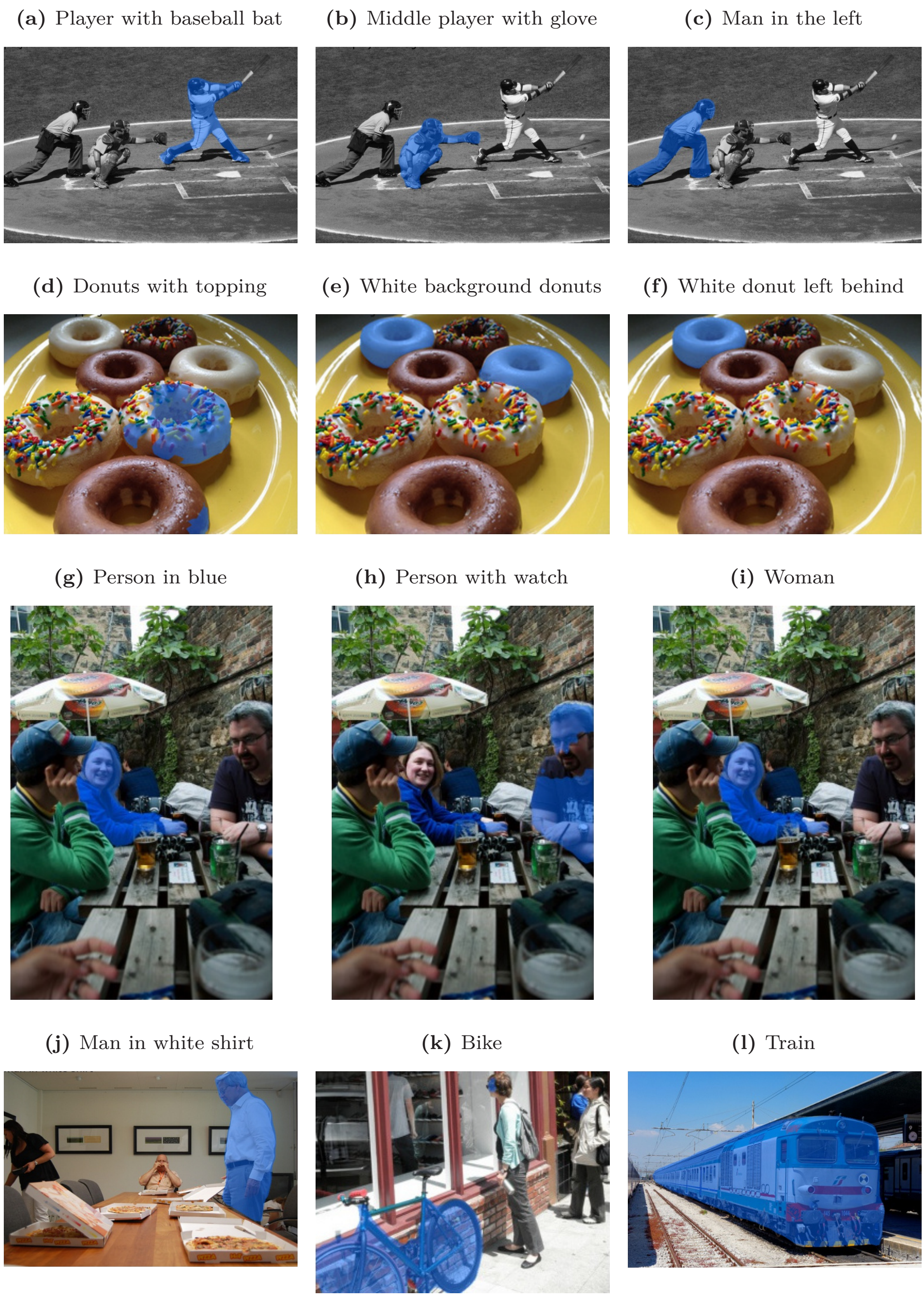
The model presented in this work behaves successfully before a great variety of images and REs. Various examples have been collected in Figure 5.1 on the next page, where the result of the segmentation on the image is shown in blue and RE used in the upper part of each figure. We can see, for example, in the first row (Figures 5.1a to 5.1c on the facing page) the same image representing a photograph taken in a baseball game, where different REs have been used successfully to refer uniquely to each of the three players that appear in the image. The segmentation obtained is correct and very precise. Furthermore, the players have been referred to in different ways: `player` and `man`, and particularized with: reference between objects (`with baseball bat` and `with glove`) and relative positioning (`in the left`).

In the second row of this same figure (Figures 5.1d to 5.1f on the next page) you can see in this case a tray with donuts and in which REs considered more complex have been used, such as `with topping`. Also, in Figure 5.1e on the facing page you can see an example of multiple selection of objects, solved successfully.

> **!** This case of multiple object segmentation is not really part of the scope of this work, despite being successfully solved by the model. It must be remembered that one of the hypotheses of Referring Expression Comprehension (REC) is that RE must be descriptive enough to refer to one—and only one—object. That is, it is assumed that the referenced object is unique.

The third and last row of this same figure collect more examples with different correctly segmented images (Figures 5.1g to 5.1l on the next page). Here they have been used as a sample RE towards different objects (e.g., `bike`, `train`). An example of

[6] Full link for "website": `https://recomprehension.com`



(a) Player with baseball bat (b) Middle player with glove (c) Man in the left

(d) Donuts with topping (e) White background donuts (f) White donut left behind

(g) Person in blue (h) Person with watch (i) Woman

(j) Man in white shirt (k) Bike (l) Train

Figure 5.1. Model evaluation succesful examples. Tested with different images and with varied REs. Figures created by the author (all). View images in color to better appreciate segmentation.

highly complex segmentation has also been shown (person with watch), in which the model works correctly despite not being too precise. In the case of figure Figure 5.1k on the preceding page it is worth highlighting as a positive point a very precise segmentation given the geometric complexity of the object. The segmentation of the object train in the Figure 5.1l on the previous page as a whole is also satisfactory (the segmentation in this case is not trivial due to the length of the object and the small size within the image of the final part of it).

Finally, within this qualitative evaluation section, we have wanted to add examples of different REs applied to the same image in Figure 5.2 on the facing page. Here we have started from the original image (Figure 5.2a on the next page) and the model has been executed with different REs, starting from the simplest to other more complex ones and in which extra reference elements have been added. As a novel contribution in this figure is the use of a RE in which differentiates between instances using a comparison: blackest cat, in Figure 5.2d on the facing page and of a RE taking into account the position of the referred object (with one leg extended).

A subjective RE (in which the referred object is not properly selected) has also been added, as a curiosity (see Figure 5.2i on the next page). In this case, it is the same model that is using the information extracted from the dataset to determine something as complex and subjective as aesthetics or beauty. Of course, this example is also outside the scope of this paper.

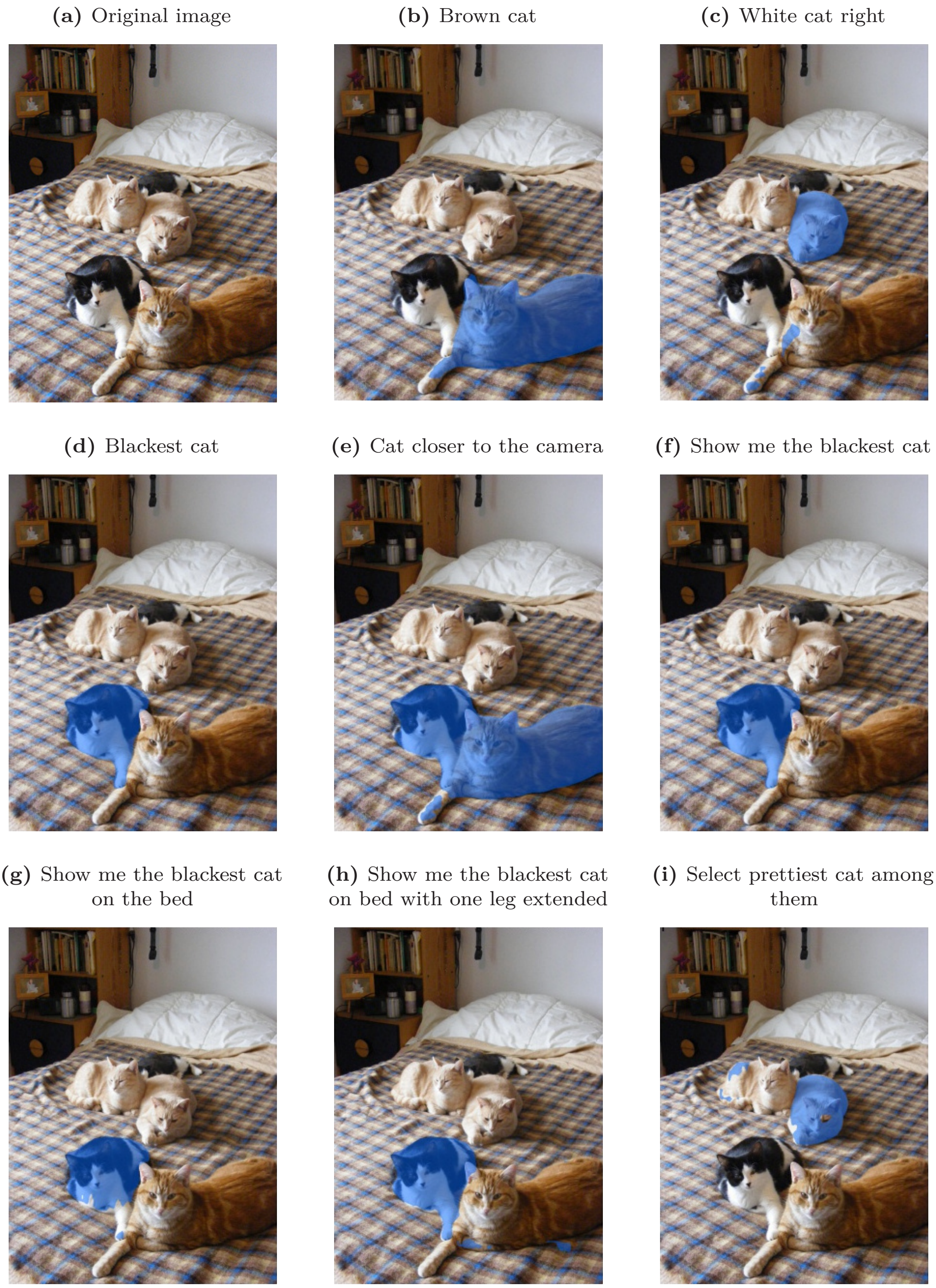
5.2.2 Study of Failed Samples

After looking at all these samples of successful comprehension in the previous section, we might think that the model is perfect. Now, unfortunately, this is not the case. Different problems appear depending on the image and the RE used, either by completely failing, or by performing an insufficiently precise segmentation or because RE is wrongly specified. Different examples of failures with this model are presented in Figure 5.3 on page 66.

Among them we show examples of imprecise segmentation (Figures 5.3a and 5.3b on page 66). In these cases the model works approximately correctly locating the referred object, but it is not capable of generating a sufficiently precise segmentation to be considered a successful sample. In other cases, furthermore, the location of the object is not even carried out correctly (Figures 5.3f and 5.3g on page 66), where it is quite possibly due to the “ignorance” of the specialized vocabulary model (as is the case of statue). This is also the case with object hair dryer in Figure 5.3d on page 66.[7] Another example of incorrect segmentation is the one shown in Figure 5.3c on page 66, but here the error is due to a bad RE wrong specified (there are multiple instances of the object being referred to).

Sometimes also, correct segmentations happen but “by chance”. This is the case of Figure 5.3d on page 66: at first we can believe that the segmentation is correct and that the model is working correctly, but it really is not. Why? The model is not able to understand the vocabulary of hair dryer (see Figure 5.3e on page 66), so it is

[7] The claim that the model “ignores” this vocabulary is conjecture by the author. Another feasible possibility in this case is that, due to the reduced size of the referred object, the segmentation is not correct.



(a) Original image
(b) Brown cat
(c) White cat right
(d) Blackest cat
(e) Cat closer to the camera
(f) Show me the blackest cat
(g) Show me the blackest cat on the bed
(h) Show me the blackest cat on bed with one leg extended
(i) Select prettiest cat among them

Figure 5.2. Comprehension results in an image with cats laying on a bed. The same image is tested with different REs. Figures created by the author (all). View images in color to better appreciate segmentation.

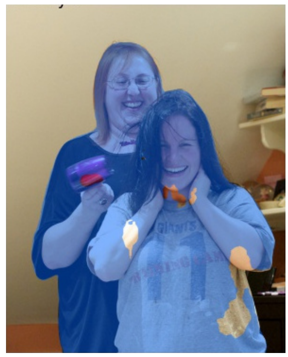
(a) Left tennis racket

(b) Blond boy looking back

(c) Banana

(d) Woman holding hair dryer

(e) Hair dryer



(f) Statue

(g) Statue of a bird

(h) Tennis match referee

(i) Tennis match referee sitting behind

Figure 5.3. Failed comprehension examples. REC task fails due to model errors, RE specification errors and lack of vocabulary. Figures created by the author (all). View images in color to better appreciate segmentation.

not really reasoning. She is simply using the word woman that she understands well and is making a conjecture as to which of the two women the RE is referring to.

Finally, in Figures 5.3h and 5.3i on the preceding page, an example is shown in which RE at first is not enough (possibly because the model is not capable of understanding the specialized word of referee), but when presenting a RE more specifically yes that the segmentation is carried out correctly. This leads one to think that—on a practical application level—it might be useful not to always segment. In other words, it would be useful to implement an extra "trust" functionality in the segmentation performed. In this way, the model could "warn" if the confidence level is not high enough. In colloquial words and using the example described: if we wanted to segment the match referee and start with RE tennis match referee, the model could warn us that the confidence it has of performing a correct segmentation is not high enough, so that we can extend this RE to provide more details to the model (tennis match referee sitting behind) and that it can segment the referred object more easily.

By visualizing information, we turn it into a landscape that you can explore with your eyes.
—David McCandless

Chapter 6

Visualization

One of the fundamental parts of this project has been to present an interactive results visualization tool focused on a user without knowledge in the field of Artificial Intelligence (AI). To do this, the creation of a web application has been chosen, mainly because of its versatility and ease of use: it does not require the installation of any specific program, only the use of a web browser.

On this website[1] you can find all the information related to this project, as well as tools to interact with the created models that solve the task of Referring Expression Comprehension (REC). All the functionality present for the user and its creation will be discussed in Section 6.1. Now, as is well condo in the field of web application design, all User Interfaces (UIs) needs code in the back end that makes it possible (see Section 6.2 on page 72 for a detailed explanation).

6.1 User Interface

The UI has been created with the idea of being as simple as possible. In this way, any type of non-specialized user will be able to carry out a qualitative evaluation of the model used. See Figure 6.1 on the following page for a screenshot of the web interface. In addition to proposing an interactive medium where REC can be done with arbitrary images and Referring Expression (RE), on the web you can also find explanations about this work, download this report and find all the source code developed. This website will be a public and transparent online version containing all the work of this thesis.

Different ways of interacting with this interface are possible. Among them, for example, three options have been added to introduce images for the realization of REC. Withing this options is that of *gallery*; different images are shown in a gallery inserted on the web. These are taken from the Common Objects in Context (COCO) dataset.[2] There is also the possibility of using a *web address*, in which an image can also be added to the web from an external source using its web address (the URL).

[1] Full link for "website": `https://recomprehension.com`
[2] It should be noted here that, when taken from the COCO dataset, most of the images that appear in this gallery have *not* been used to train the neural model.

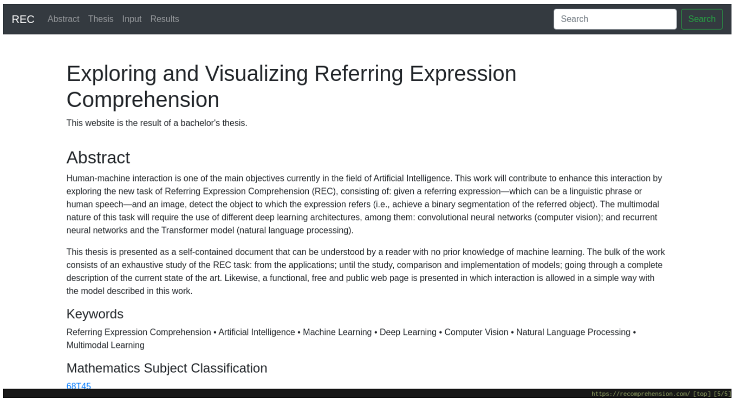


Figure 6.1. Website screenshot. In it you can see the navigation bar and the first part of the web. Figure created by the author.

Finally, it is possible to use the *local* option, in which you can upload an image from your own computer's local storage to the web.

In the case of adding RE there are two options in order to facilitate interaction with the model. These are the following: keyboard and voice. Firstly, the RE can be entered using the *keyboard* in the usual way. But also it is possible to enter RE more comfortably using your own *voice*. To do this, it is only necessary to press the corresponding button on the main page so that the window corresponding to the voice input of commands opens (see Figure 6.2). Here we will have to give permissions to the web to access the microphone and we will be able to record the RE that we want.

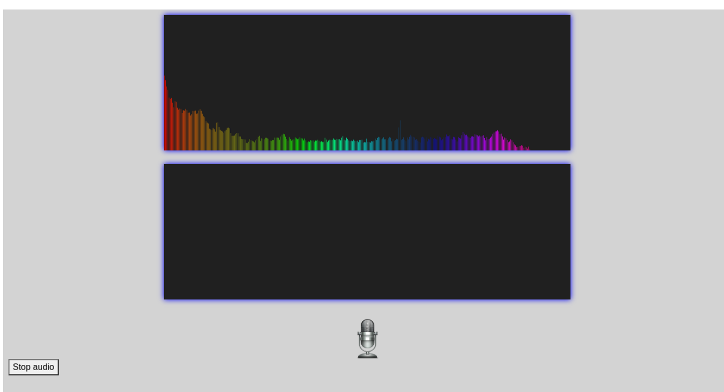


Figure 6.2. Web interface for voice input. Here you can speak directly into the computer's microphone as an alternative method of entering a RE. This audio will be converted using a neural model to text. Figure created by the author.

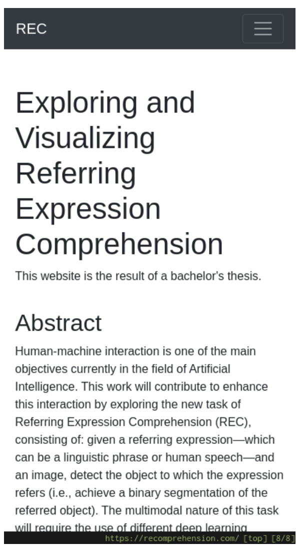
6.1.1 Responsive Design and Accessibility

Responsive web design and accessibility are two different concepts, but they are still related to each other in many ways. These two concepts fit within the idea of User Experience (UX) which is the way in which a user interacts with a certain product or service. This includes the user's perception of efficiency, ease of use and usefulness.

Responsive design is concerned with providing the user with the best possible viewing experience regardless of the device being used. That is, the interface of a website is capable of adapting to the dimensions of the screen being used: the website will be displayed differently depending on whether it is being used on a computer, tablet or mobile monitor. Accessibility is concerned with ensuring that the content is easily usable, navigable by people with certain disabilities (e.g., vision problems).

We can see how the web created adapts to different widths by comparing Figure 6.1 on the facing page with Figure 6.3. Among other aspects, the gallery adapts the number of columns according to the device and the navigation bar expands or collapses also depending on the width of the screen. In addition to these two commented elements (navigation bar and gallery) other different elements also adjust to the different screen widths. Among them the footer, the width of the resulting segmented image, the size of the RE and many more.

(a) Collapsed navigation bar.



(b) Expanded navigation bar.

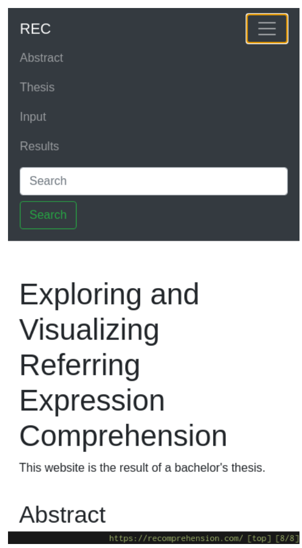


Figure 6.3. Responsive web design visualization. The web is displayed in a typical mobile device width. The web is designed to be displayed correctly regardless of the width of the device used. Figure created by the author (both).

Another fundamental aspect when making a web design is that of *accessibility*. With the term accessibility we refer to that all kinds of people could be accessed regardless of their disabilities. That is, we try to keep the whole population as possible in mind as possible users and guarantee access and use of the web. Berners-Lee

[Ber21][3], World Wide Web Consortium (W3C)[4] Director and inventor of the World Wide Web, gives a very interesting affirmation of accessibility within the scope of web development, where he defends the right of all people without ingrousing their disability.

> *The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.*
>
> —Berners-Lee [Ber21]

To guarantee the accessibility of this website, Accessible Rich Internet Applications (ARIA) has been used, which is a specification of W3C that specifies how to increase the accessibility of web pages. This has been facilitated by the use of a style library Cascading Style Sheet (CSS) called Bootstrap[5].

6.1.2 Guided Usage Example

To carry out the task of REC on an image, it is only necessary to carry out these three steps,

1. **Choose image.** As we have already discussed at the beginning of Section 6.1 on page 69, three ways are available: select from the gallery, add the web address or choose it from the local storage of the computer.

2. **Enter RE.** Likewise, for the introduction of RE there are two methods: by using the keyboard and by using voice.

3. **View results.** Finally, after clicking on the button `Submit`, we can see the result of the execution of the program.

These methods have been tested and all of them work correctly, although with different execution times depending on the method used. This execution time will mainly depend on three factors, server usage, selected image quality and the input method of RE (by voice it will take longer).

> To have an approximation, using images from the gallery shown on the web, with a single user browsing (unsaturated server, which is usual) and entering the RE using the keyboard, the complete execution of the program is about 5 seconds.

6.2 Back End

The main functionality offered by this website is to be able to interact with this work, i.e., to be able to execute the present model to perform REC. This entails the

[3] Tim Berners-Lee. *Introduction to Web Accessibility — Accessibility in Context*. `https://www.w3.org/WAI/fundamentals/accessibility-intro/`. [Online; accessed 6 April of 2021]. 2021.
[4] Is the main international standards organization for the World Wide Web
[5] Full link for "Bootstrap": `https://getbootstrap.com/`

execution of code, which could be chosen to execute it in two different places: on the user's computer (JavaScript should be used) or on the web server (the back end).

Due to the high use of computational resources in the execution of the model, it has been decided to carry it out completely in the back end.[6] For this, a Application Programming Interface (API) (created in PHP) has been created to facilitate communication between front end and back end without having to reload the page. We have two main routes that we will use to communicate the web interface with the backend. These are the following,

- **REC.** Internally called `api/comprehend.php`, which allows to perform the task of REC by calling the appropriate Python files internally on the server (executing the segmentation model on user input, see Section 4.1 on page 45.) It takes into account the different ways in which the image and RE have been added.
- **Speech to Text (STT).** Internally called `api/stt.php`, which allows to perform the task of converting between speech and text. Run the Silero model (see Section 4.2 on page 54) to convert the audio recorded with the microphone by the user to text.

Here we will see the structure of the back end. An explanatory graph is found in Figure 6.4 on the next page. As you can see, the server architecture is divided into two parts, the front end part and the backend part. It is in the first instance the client (user) who, using his browser, accesses the web address. Once the request is received, the web server returns the generic request to the user: that is, the `index.html` along with all its corresponding CSS style sheets and JS code files. The client will then interact as desired with the different elements present in the interface. When you decide to add audio or perform the REC task, it will be a JS file that responds to your request through the API of the server. This API is in charge of understanding, processing and responding to the user's request, by executing the corresponding Python code.

[6] This also facilitates the possibility of executing the code in Python, since if it would not be necessary to port all the code to JavaScript or use libraries with which to emulate Python within the client's browser.

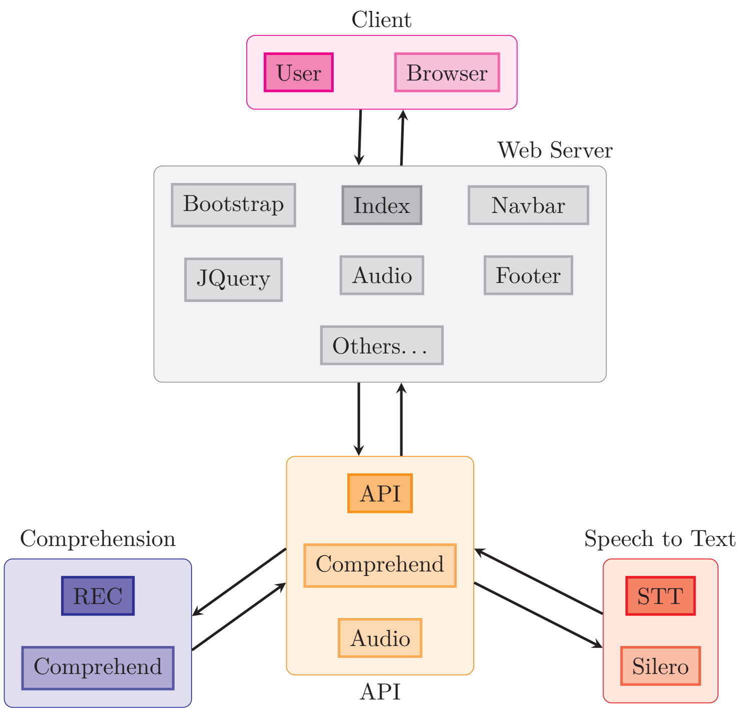


Figure 6.4. Program architecture. You can observe the transmission of data from the client (the user with his web browser), up to the generation of results through the API on the server. Figure created by the author.

Give me six hours to chop down a tree and I will spend the first four sharpening the axe.
—Abraham Lincoln

Chapter 7

Project Analysis

The analysis of a project is fundamental from an engineering point of view. This analysis falls within the scope of project management, which constitutes the area in charge of managing the evolution of the project, controlling and responding to problems that appear and facilitating its completion and approval. Here we will analyze the work carried out from a resource management point of view in terms of planning and scheduling the tasks (see Section 7.1), an analysis of the cost of the project will be carried out (see Section 7.2 on the next page) and, finally, the environmental impact will be studied (see Section 7.3 on page 79).

7.1 Planning and Scheduling

This project will (obviously) entail carrying out a series of activities for its development. This time *scheduling* of these activities and everything related to the consideration of the necessary resources are the most important functions to develop in project *planning*.

The main objective of *planning* is to obtain a distribution of activities over time and tries to use resources in a way that minimizes the cost of the project, always complying with the different conditions required: start/end date, available technology, available resources, the maximum possible level of occupation of these resources, etc. That is, project planning consists of a scheduling of activities and a management of resources—which can be material or human—to obtain a cost objective complying with the conditions imposed/demanded by a particular client.

7.1.1 Table of Activities

The scheduling of activities will allow us to have a project execution calendar where the start and end dates of the different activities in which the project has been decomposed are reflected. To facilitate understanding of the different activities, they have been divided into 5 large groups:

- **Learn basics of Machine Learning (ML) and Deep Learning (DL).** This set of tasks has consisted of acquiring basic knowledge in the areas of

ML/DL that have allowed us to continue in the project. Much of what is learned here is precisely what is described in Chapter 2 on page 9.

- **Learn thesis topic.** Once the basic knowledge was established in DL, we have proceeded to go deeper into advanced knowledge and more related to the specific field of the thesis. For this, different recommended papers have been read and the existing literature about state-of-the-art models has been read.
- **Models creation.** This set of tasks coincides with Chapter 4 on page 45. This is where the two models used in this work are presented.
- **Web development.** Here all the activities related to the development of the web are collected (see Chapter 6 on page 69). From learning front end languages to publishing the web with your own domain and going through back end programming.
- **Bachelor's thesis.** These tasks correspond to those requested by the university: writing the work report, and creating and preparing the final presentation.

The set of activities broken down is shown in Table 7.1 on the facing page, where you can see in detail the tasks that make up the main activities. Likewise, approximate start and end dates are shown for both the tasks and the main activities.

7.1.2 Gantt Chart

The information collected in the form of an activity table in the previous section (Section 7.1.1 on the previous page) can be shown more graphically with a diagram. The best known tool to represent the planning of tasks over time is the one created by Gantt [Gan73][1]. This diagram, named in honor of its creator as the Gantt chart, is a graphical tool whose objective is to expose the time of dedication planned for different tasks or activities over a given total time.

For this specific work, the corresponding Gantt chart is shown in Figure 7.1 on page 78. This chart is exactly the graphical representation of the distribution of tasks in Table 7.1 on the next page.

7.2 Cost Analysis

The total cost associated with this project is divided into two parts: the personal cost and the infrastructure cost.

Personal Cost

Regarding personal cost, it refers to the number of hours dedicated to carrying out this work, including all its parts. That is, here they will be considered from the hours dedicated to learning, such as the hours dedicated to programming, such as the hours

[1] Henry Laurence Gantt. *Work Wages and Profits (Management in History No 41)*. Hive Publishing Company, Sept. 1973. isbn: 0879600489.

Table 7.1. Main activities broken down into tasks and with approximate start and end dates. Note that various tasks have been carried out in parallel. Table created by the author.

Code	Activity	Start	End
A	**Learn basics of ML/DL**	Oct.	Jan.
A1	ML course [Ng20]	-	-
A2	DL lectures from UPC [Gir20]	-	-
A3	Stanford CS231n: CNNs for Visual Recognition [LKX20]	-	-
A4	DL specialization [NKM20]	-	-.
B	**Learn thesis topic**	Dec.	Feb.
B1	Multimodal learning lectures [Gir20]	-	-
B2	Publications	-	-
B3	State-of-the-art papers on REC	-	-
C	**Models creation**	Jan.	Apr.
C1	Server usage	-	-
C2	Multiple iterations	-	-
C3	Generate test values	-	-
D	**Web development**	Feb.	Apr.
D1	Front end (HTML, CSS, JS)	-	-
D2	API creation (PHP)	-	-
D3	Web server configuration	-	-
D4	Publish website (domain, server)	-	-
E	**Bachelor's thesis**	Dec.	May
E1	Write thesis (LaTeX)	-	-
E2	Create presentation slides (LaTeX)	-	-
E4	Prepare presentation	-	-

Note. Start and end dates shown are approximate.

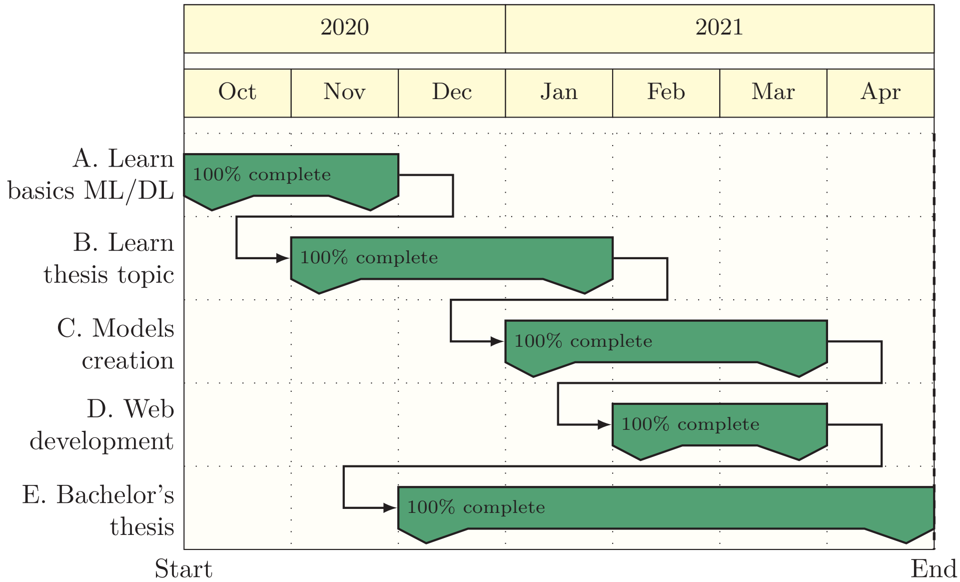


Figure 7.1. Gantt chart of main activities. The duration and relationship between main activities is shown graphically. Figure created by the author.

dedicated to web design and the hours dedicated to the writing of the memory and the creation of the presentation of this work.

To estimate the hours dedicated, we will use European Credit Transfer and Accumulation System (ECTS), which is a standard for comparing academic credits. As is known, one credit ECTS is equivalent to a dedication of 25–30 hours. In this case, as we are doing a bachelor thesis of two degrees, we will add the credits allocated to each degree. Specifically, for the degree in Industrial Technology Engineering, there are 12 credits and for the degree in Mathematics there are 15 credits. In total 27 ECTS credits.

Therefore, using the equivalence of 1 credit ECTS with 27.5 hours, we have that the number of hours dedicated to work will be,

$$27\,\text{ECTS credits} \times \frac{27.5\,\text{h}}{1\,\text{ECTS credit}} = 742.5\,\text{h}, \tag{7.1}$$

which, assuming a wage of 12 €/hour, makes a total cost *personal* of,

$$742.5\,\text{h} \times \frac{12\,€}{1\,\text{h}} = 8910\,€. \tag{7.2}$$

! It must be taken into account that this estimate of 12 €/hour is an *aproximation*, in order to obtain data on the personal cost money. Of course, the number of hours dedicated to training will have a much lower remuneration than that of the hours dedicated to the creation of the model and to the remuneration of the hours dedicated to web design.

Infrastructure Cost

Regarding the *infrastructure* cost, it will only be necessary to include the expenditure made on servers, since the rest of the tools used are free (as in *freedom*) software, but also free in terms of price.[2] The servers used for training have been assigned by Vector Institute[3]. The server used for the web has an approximate cost of 20 € / month and has been rented for a total of 2 months. In total 40 €.

Total Cost

Therefore, the total cost can be calculated by adding the two cost sources, personnel and infrastructure. Clearly, the personal cost far exceeds the infrastructure cost (mainly because the cost of the training servers with Graphics Processing Unit (GPU) has been zero as they have been provided free of charge). The total cost was 8950 €.

7.3 Environmental Impact

The environmental impact that this work has produced is minimal, since it has been a software development. The only element that makes sense to consider in this regard is the use of electrical energy to power the computer and servers, since the generation of this electrical energy will lead to certain emissions of CO_2.

Assuming an approximate average consumption of the computer of 150 W, and that it has been used during the total of 742.5 h that the project has lasted (see Section 7.2 on page 76), we have that, at an energy level, they have been consumed,

$$150\,\mathrm{W} \times 742.5\,\mathrm{h} = 111.375\,\mathrm{kWh}. \tag{7.3}$$

We can now, using the online emission calculator of CO_2, created by Government of Aragon (Spain) [Gov21][4], conclude that the emissions of CO_2 are 39 kg of CO_2.

> These emissions of CO_2 are those that a single gasoline car would emit during a journey of 200 km (from [Gov21]).

We could also consider the emissions of CO_2 due to the use of servers during training and the web server. Now, in the first case, it is difficult to quantify, since it is a multi-node server with users. And, in the second case, it is difficult to quantify the use of the web server, since it is open to the public and depends on the number of users entering the web.

In any case, unsurprisingly, the environmental impact of this project is minimal.

[2] Here they enter the use of Python, PyTorch for modeling; the HTML, CSS and JS languages for the creation of the web interface; PHP for Application Programming Interface (API); Apache as a web server; and LaTeX for the writing of the report and the creation of the presentation.

[3] Full link for "Vector Institute": `https://vectorinstitute.ai/`

[4] Government of Aragon (Spain). *CO_2 Emission Calculator*. `http://calcarbono.servicios4.aragon.es/`. [Online; accessed 6 April of 2021]. 2021.

Now this is not the end.
It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.
—Winston Churchill

Chapter 8

Conclusions

Carrying out this research work within the university framework of a bachelor's thesis has allowed me, on a personal level, to initiate and delve into topics related to Machine Learning (ML) that are currently on everyone's lips. The Artificial Intelligence (AI) has come to stay in our current society. The specific topic of this work Referring Expression Comprehension (REC) has allowed me to work in the field of multimodal learning, so that I have been able to explore at the same time the fields of Computer Vision (CV) and Natural Language Processing (NLP), which for me were a novelty. Being able to start from scratch and finish training and modifying state-of-the-art models produces in me a satisfying feeling on an academic level.

Likewise, it has allowed me to improve my skills in the field of programming: both in the development of neural models using the Python PyTorch library, and in the field of web development and the creation of Application Programming Interfaces (APIs) and management of servers. Within this improvement of programming skills, there is also the use of professional servers for training neural models using GPUs, for which it is necessary to use specialized software such as Slurm used as an open-source job scheduler.

Furthermore, I am satisfied with the results obtained, despite not having obtained significant improvements in the Referring Expressions for Video Object Segmentation (RefVOS) model. Having been able to train professional models and being able to modify it and understand all the small parts that make it up are already a source of joy for me. In addition, being able to provide the general public with a website where they can easily interact with models so complex that the one presented, I think is positive for society in general. Tools similar to this one may be useful for future researchers in this or similar field.

Also, this project has allowed me to grow as a person. It has been carried out in a turbulent time within the COVID-19 pandemic, which has forced remote work. Having to work remotely with a large research laboratory in AI in another time zone is not an easy task. I appreciate all the help received by email and by video conference from my thesis supervisors and the Vector Institute members with whom I have been fortunate to discuss aspects of the work. Additionally, writing this thesis, such an extensive document on a personal level, has allowed me to grow as a student, there have been many decisions that have had to be made along the way in relation to the writing of this work.

8.1 Future Work

In order to advance in the task of REC it is essential to first know the main limitations that currently exist. One of the main problems that arise is the difficulty of understanding what the model is doing. All the models present in the current literature present some method in which the embeddings of Referring Expression (RE) and the image are joined. Now, this process is currently a “black box” for researchers: the reasoning process of the model cannot be visualized. This greatly penalizes the possibilities of improving the models, since it makes it very difficult to interpret the decisions of the model in the understanding process.

In addition, this lack of interpretability of the reasoning method of the model is enhanced by the evaluation process in which only the final prediction is taken into account, so that a concrete evaluation of the reasoning process is not made step by step. That is, the evaluation metrics used in current state-of-the-art studies are not capable of extracting useful information about the real reasoning capacity of the model and, therefore, do not provide a vision about the deficiencies of the model.

Another important limitation in the REC task is the lack of quality dataset for training. It has been possible to manually observe samples from the dataset that are not adequate (RE misspelled, containing bad words, etc.). In addition, there is a clear imbalance in the samples of the current datasets. Most of the RE present refer to the objects in the image using attributes. This imbalance can lead to models where there is no deep reasoning process, in which segmentation is only learned depending on the class to which the object belongs. It could even be the case that the models ignore RE and only make a random guess of the most representative object (that is, using only the information present in the image).

Following the current jobs path may not lead to significant improvements in model performance. That is, adding complexity (increasing the number of basically trainable parameters) to current techniques REC may not be the way to go. Designing more sophisticated models but under the same current principles will not necessarily lead to significant improvements in the task of REC. To achieve significant improvement, the next logical step is to try to find models in which some sophisticated method of reasoning can be exploited more effectively. I consider that the most successful ones in the future would be multi-step reasoning models, in which relevant information is actually extracted from RE. In addition, it would be very useful if each of these reasoning steps could be visualized and validated with objective metrics.

A simple example of these multi-step reasoning would be for example with the following RE, `woman in red dress sitting on the right` and an image with a large group of people. An ideal multi-step reasoning model would work as follows:

1. Find all the women present in the image, these objects will be the only solution candidates.

2. From these women choose all those who wear a red dress.

3. From this group select those that are seated.

4. Finally, if there is more than one possibility, select the one on the right.

In this type of model, or similar, the real reasoning would be guaranteed and it would be easy to evaluate step by step.

In the case of datasets, they could also be improved. It would be necessary to collect more data, of higher quality and with different types of RE. In addition, it would be very useful for the training and validation of the models to have a metric to evaluate the difficulty of one RE. This dataset expansion could be done, if necessary, making use of generative models, i.e., synthetic data could be used. This would be especially useful to correct already detected imbalances.

Also add that you can see how to apply these models to video in addition to image. The model presented in this thesis, obviously, could also be used for video using it frame by frame. Now, the temporal relationship between different frames would be neglected. Here it would be of vital importance to ensure the efficiency and speed of the models used, to make real-speed comprehension possible.

Other possibilities such as future lines of research within this work, but which are outside the scope of a student (they would be more focused on an institution or university) would be those related to the dataset. That is, extend the existing one and clean it of errors, of which it is quite full (there are too many REs of poor quality). In addition, for a leading institution in this area, the possibility of creating an objective classification table for the classification of models within this task could be considered. This is something that does not currently exist and would be very useful to be able to make more technical comparisons. This table could collect all current modelsf with an extensive set of evaluation metrics, allowing future researchers to quickly get an idea of the current state of the art.

Organizing files is like organizing your room:
it should be clean and easy to navigate through.
—George Sun

Appendix A

File Structure

During the development of this work, different code files have been created and used in different programming languages. Establishing a consistent, logical, clear and easy-to-navigate file structure has been a critical element in facilitating the creation and editing of these files. In addition, Git has been used as a version control system for tracking changes in the text and code files. If the reader wishes, he can consult all the files in the project repository[1]. Below is the first level main structure.

```
Bachelor's Thesis
├── Code
├── Datasets
├── Thesis
├── Website
├── .gitignore
└── README.org
```

The total set of files is divided into 4 large groups that will be explored individually: the code directory (see Appendix A.1), the datasets directory (see Appendix A.2 on the following page), the directory containing the code of this thesis (see Appendix A.3 on page 87) and the website directory (see Appendix A.4 on page 88).

A.1 Code

Inside this directory you will find all the code related to the implementation of the model used, as well as files for the train and test of the model. Also, here you will find the files for reading the dataset and those that are executed in the backend of the server (they are called by the Application Programming Interface (API) in PHP). The files used for the iterations of the different models are also collected here (some of these files should possibly be retrieved from the Git history).

```
Code
└── .venv
```

[1] Full link for "repository": `https://gitlab.com/david-alvarez-rosa/bachelor-thesis`

```
│   └── * Python virtual environment
├── checkpoints
│   ├── model.pth
│   └── * Possibly other checkpoints
├── lib
│   ├── utils.py
│   ├── deeplabv3_emb.py
│   └── segmentation.py
├── .gitignore
├── args.py
├── compehend.py
├── dataset.py
├── model.py
├── README.org
├── refer.py
├── test.py
├── train.py
├── transforms.py
└── utils.py
```

A.2 Datasets and Utils

The datasets used in their original format will be stored in this directory.

```
Datasets
├── refcoco
│   ├── images
│   ├── annotations.json
│   ├── refs(google).p.py
│   └── refs(unc).p.py
├── refcoco+
│   └── * Similar to refcoco ↑
├── refcocog
│   └── * Similar to refcoco ↑
└── README.org
```

The *utils* directory containing different files useful for various functions. Among them are the shell executables to synchronize files with the remote server for training and with the web server. Slurm[2] configurations files used are also present here.

```
Utils
├── .gitignore
├── createSmall.py
└── getImgNames.py
```

[2] Slurm is an free and open-source job scheduler that is used in the servers provided by VectorInstitute.

```
│
├── launch
├── newServer.sh
├── testAPI.sh
├── vector.sh
└── vectorInteractive.sh
```

A.3 Thesis

This directory contains all the files that this thesis typed with LaTeX makes possible: source code, images, vector graphics, acronyms, references, etc.

```
Thesis
├── Appendices
│   └── Appendix{A,B,C}.tex TeX
├── Build
│   └── * Externalized build files
├── Chapters
│   └── Chapter{1,2,...,8}.tex TeX
├── Figures
│   ├── Images
│   │   └── * Document images
│   ├── Logos
│   │   └── * University logos
│   ├── Icons
│   │   └── * Program icons
│   ├── SVGs
│   ├── * Document SVGs
│   └── Tikz
│       └── * Vector graphics
├── FrontBack
│   ├── FrontMatter.tex TeX
│   └── BackMatter.tex TeX
├── Utils
│   ├── Acronyms.tex TeX
│   ├── EasterEgg.tex TeX
│   ├── Index.ist
│   ├── Preamble.tex TeX
│   └── References.bib BibLaTeX
├── .gitignore
├── README.org
├── Slides.tex TeX
└── Thesis.tex TeX
```

A.4 Website

This directory is an exact replica of the one on the web server, and contains all the files necessary for its operation.

```
Website
├── api
│   ├── .cache
│   │   └── * PyTorch cache files
│   ├── audio
│   │   └── * Temp audio files
│   ├── results
│   │   └── * Temp output images
│   ├── Code -> ../../Code
│   ├── comprehend.php
│   └── upload_wav.php
├── css
│   ├── audio.css
│   ├── bootstrap.min.css
│   └── main.css
├── img
│   └── * Website images and icons
├── js
│   ├── recorder
│   │   ├── recorder.js
│   │   └── recorderWorker.js
│   ├── audio.js
│   ├── audiodisplay.js
│   ├── bootstrap.bundle.min.js
│   ├── imgFileNames.js
│   ├── jquery-3.5.1.slim.min.js
│   └── main.js
├── .gitignore
├── .htaccess
├── datasets -> ../Datasets
├── favicon.ico
├── index.html
├── README.org
├── site.webmanifest
└── thesis.pdf
```

Note that there is a symbolic link here pointing to the datasets directory and another pointing to the code directory. Therefore, all these directories are actually part of the web server.

Talk is cheap. Show me the code.
—Linus Torvalds

Appendix B

Implementation Details

The most representative details about the implementation and programming of this project will be collected in this chapter. The code files that collect the general ideas used in this work will be exposed verbatim, but various existing auxiliary files will not be shown here to avoid being extended too much. The curious reader can consult the entire bulk of the code used and its evolution in the official repository[1] of this project (also consult web[2] for more information). This chapter will divide itself into the code files—mainly Python model implementation —in Appendix B.1, the web server-related implementation (see Appendix B.2 on page 107), and files related to using servers (see Appendix B.3 on page 116).

B.1 Code Files

Among the different files used for the implementation, training and testing of the model are those shown below. Many more than these files have been used, since the model creation process has been an iterative process, in which various modifications of the base model have been tested.

Below is the code used to carry out the testing of the model.

../Code/test.py

```python
"""File for testing the model

This file can evlauate the model in different datasets and computes the metrics
related to the Jaccard Index (also called Intersection over Union) and also
accuracy or Precision at Threshold (such as, for instances, Prec@0.5).
"""

import time
import numpy as np
import torch
from transformers import BertModel
```

[1] Full link for “official repository”: `https://gitlab.com/david-alvarez-rosa/bachelor-thesis`
[2] Full link for “web”: `https://recomprehension.com`

```python
import transforms
from lib import segmentation
from dataset import ReferDataset
import utils
from model import Model


def evaluate(data_loader, model, device, dataset=None, results_dir=None):
    """Evaluate the model in the given dataset."""

    model.eval()

    loss_value = 0
    cum_intersection, cum_union = 0, 0
    jaccard_indices = []

    tic = time.time()

    for imgs, targets, sents, attentions, sent_ids in data_loader:
        print(time.time() - tic)
        tic = time.time()

        imgs, attentions, sents, targets = \
            imgs.to(device), attentions.to(device), \
            sents.to(device), targets.to(device)

        sents = sents.squeeze(1)
        attentions = attentions.squeeze(1)

        with torch.no_grad():
            outputs = model(sents, attentions, imgs)
            loss = torch.nn.functional.cross_entropy(outputs, targets,
                                                     ignore_index=255)
            masks = outputs.argmax(1)

        loss_value += loss.item()

        jaccard_indices_batch, intersection, union = \
            utils.compute_jaccard_indices(masks, targets)
        jaccard_indices += jaccard_indices_batch
        cum_intersection += intersection
        cum_union += union

        if results_dir is not None:
            utils.save_output(dataset, sent_ids, masks, results_dir)

        # Release memory.
        del imgs, targets, sents, attentions, sent_ids

        # Added.
        print("loss: {:.4f}".format(loss_value/len(data_loader)))
        print("len_jaccard_indices: ", len(jaccard_indices))
        mean_jaccard_index = np.mean(np.array(jaccard_indices))
        print("Mean IoU is {:.4f}.".format(mean_jaccard_index))
        print("Overall IoU is {:.4f}.".format(cum_intersection/cum_union))
        print("\n\n")

    print("\n"*10)
```

```
    print("loss: {:.4f}".format(loss_value/len(data_loader)))
    print("jaccard_indices: ", jaccard_indices)
    mean_jaccard_index = np.mean(np.array(jaccard_indices))
    print("Mean IoU is {:.4f}.".format(mean_jaccard_index))
    print("Overall IoU is {:.4f}.".format(cum_intersection/cum_union))


def main(args):
    # Define dataset.
    dataset = ReferDataset(args,
                           split=args.split,
                           transforms=transforms.get_transform())

    data_loader = torch.utils.data.DataLoader(dataset,
                                              batch_size=args.batch_size,
                                              num_workers=args.workers,
                                              collate_fn=
                                              utils.collate_fn_emb_berts)

    # Segmentation model.
    seg_model = segmentation.deeplabv3_resnet101(num_classes=2,
                                                 aux_loss=False,
                                                 pretrained=False,
                                                 args=args)

    # BERT model.
    bert_model = BertModel.from_pretrained(args.ck_bert)

    # Load checkpoint.
    checkpoint = torch.load(args.resume, map_location="cpu")
    bert_model.load_state_dict(checkpoint["bert_model"], strict=False)
    seg_model.load_state_dict(checkpoint["model"], strict=False)

    # Define model and sent to device.
    model = Model(seg_model, bert_model)
    device = torch.device(args.device)
    model.to(device)

    evaluate(data_loader=data_loader,
             model=model,
             device=device,
             dataset=dataset,
             results_dir=args.results_dir)


if __name__ == "__main__":
    from args import get_parser
    parser = get_parser()
    main(parser.parse_args())
```

Below is the code used for training the different versions of the model (keep in mind that different parts of files are parameter dependent). Actually multiple versions of this same file have been used, since different parts of it have been modified.

../Code/train.py

```
"""File for testing the model

This script has been used for model training.
"""

import time
import torch
from functools import reduce
import operator
from transformers import BertModel
from lib import segmentation
import transforms
import utils
import gc
from dataset import ReferDataset
from model import Model
import test


def adjust_learning_rate(optimizer, epoch, args):
    """Sets the learning rate to the initial LR decayed by 10 every 30
    epochs"""
    lr = args.lr - args.lr_specific_decrease*epoch
    for param_group in optimizer.param_groups:
        param_group["lr"] = lr


def train_epoch(model, optimizer, data_loader, lr_scheduler, device, epoch,
                args):
    """Train and compute loss and accuracy on train dataset_train.
    """

    model.train()

    for imgs, targets, sents, attentions, sent_ids in data_loader:
        # Sent data to device.
        imgs, attentions, sents, targets = \
            imgs.to(device), attentions.to(device), \
            sents.to(device), targets.to(device)
        sents = sents.squeeze(1)
        attentions = attentions.squeeze(1)

        # Compute model output and loss.
        outputs = model(sents, attentions, imgs)
        loss = torch.nn.functional.cross_entropy(outputs, targets,
                                                 ignore_index=255)

        # Backpropagate.
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # Adjust learning rate.
        if args.linear_lr:
            adjust_learning_rate(optimizer, epoch, args)
        else:
```

```
            lr_scheduler.step()

        # Release memory.
        del imgs, targets, sents, attentions, loss, outputs
        gc.collect()
        torch.cuda.empty_cache()


def evaluate_epoch(results_dir, device, model, loader_train, loader_val,
                   dataset_val):
    # Evaluate in train dataset.
    print("--- Train ---")
    test.evaluate(data_loader=loader_train,
                  model=model,
                  device=device)

    # Evaluate in validation dataset.
    print("\n--- Validation ---")
    test.evaluate(data_loader=loader_val,
                  model=model,
                  device=device,
                  dataset=dataset_val,
                  results_dir=results_dir)


def new_epoch(model, optimizer, dataset_train, dataset_val, loader_train,
              loader_val, lr_scheduler, device, epoch, args):
    time_start_epoch = time.time()

    # Train.
    time_start_train = time.time()
    train_epoch(model=model,
                optimizer=optimizer,
                data_loader=loader_train,
                lr_scheduler=lr_scheduler,
                device=device,
                epoch=epoch,
                args=args)
    time_end_train = time.time()

    # Evaluate.
    time_start_evaluate = time.time()
    evaluate_epoch(results_dir=args.results_dir + str(epoch+ 1) + "/",
                   device=device,
                   model=model,
                   loader_train=loader_train,
                   loader_val=loader_val,
                   dataset_val=dataset_val)
    time_end_evaluate = time.time()

    # Times.
    time_end_epoch = time.time()

    print("\n--- Time ---")
    print("time_train: {:.2f}s".format(time_end_train - time_start_train))
    print("time_evaluate: {:.2f}s".format(time_end_evaluate -
                                          time_start_evaluate))
```

```
    print("time_epoch: {:.2f}s".format(time_end_epoch - time_start_epoch))


def main(args):
    device = torch.device(args.device)

    # Train dataset.
    dataset_train = ReferDataset(args=args, split="train",
                                 transforms=transforms.get_transform(train=
                                                                    True))

    sampler_train = torch.utils.data.RandomSampler(dataset_train)
    loader_train = torch.utils.data.DataLoader(
        dataset=dataset_train,
        batch_size=args.batch_size,
        sampler=sampler_train,
        num_workers=args.workers,
        collate_fn=utils.collate_fn_emb_berts,
        drop_last=True)

    # Validation dataset. Modified.
    dataset_val = dataset_train
    # dataset_val = ReferDataset(args=args,
    #                            split="val",
    #                            transforms=transforms.get_transform())
    sampler_val = torch.utils.data.SequentialSampler(dataset_val)
    loader_val = torch.utils.data.DataLoader(
        dataset=dataset_val,
        batch_size=1,
        sampler=sampler_val,
        num_workers=args.workers,
        collate_fn=utils.collate_fn_emb_berts)

    # Segmentation model.
    seg_model = segmentation.deeplabv3_resnet101(num_classes=2,
                                                 aux_loss=False,
                                                 pretrained=False,
                                                 args=args)

    # BERT model.
    bert_model = BertModel.from_pretrained(args.ck_bert)

    # Load checkpoint.
    device = torch.device(args.device)
    checkpoint = torch.load(args.resume, map_location=device)

    bert_model.load_state_dict(checkpoint["bert_model"], strict=False)
    seg_model.load_state_dict(checkpoint["model"], strict=False)

    # Define model and sent to device.
    model = Model(seg_model, bert_model)

    model.to(device)

    params_to_optimize = [
        {"params": [p for p in seg_model.backbone.parameters()
                    if p.requires_grad]},
```

```
        {"params": [p for p in seg_model.classifier.parameters()
                    if p.requires_grad]},
        # the following are the parameters of bert
        {"params": reduce(operator.concat,
                          [[p for p
                            in bert_model.encoder.layer[i].parameters()
                            if p.requires_grad] for i in range(10)])},
        {"params": [p for p in bert_model.pooler.parameters()
                    if p.requires_grad]}
    ]

    if args.aux_loss:
        params = [p for p in seg_model.aux_classifier.parameters()
                  if p.requires_grad]
        params_to_optimize.append({"params": params, "lr": args.lr * 10})

    optimizer = torch.optim.SGD(
        params_to_optimize,
        lr=args.lr, momentum=args.momentum, weight_decay=args.weight_decay)

    if args.fixed_lr:
        lr_scheduler = torch.optim.lr_scheduler.LambdaLR(
            optimizer,
            lambda x: args.lr_specific)
    elif args.linear_lr:
        lr_scheduler = None
    else:
        lr_scheduler = torch.optim.lr_scheduler.LambdaLR(
            optimizer,
            lambda x: (1 - x / (len(loader_train) * args.epochs)) ** 0.9)

    t_iou = 0

    if args.resume:
        optimizer.load_state_dict(checkpoint["optimizer"])

        if not args.fixed_lr:
            if not args.linear_lr:
                lr_scheduler.load_state_dict(checkpoint["lr_scheduler"])


    for epoch in range(args.epochs):
        print(("\n" + "="*25 + " Epoch {}/{} " + "="*25).format(epoch + 1,
                                                                 args.epochs))
        new_epoch(model=model,
                  optimizer=optimizer,
                  dataset_train=dataset_train,
                  dataset_val=dataset_val,
                  loader_train=loader_train,
                  loader_val=loader_val,
                  lr_scheduler=lr_scheduler,
                  device=device,
                  epoch=epoch,
                  args=args)

        # Only save if checkpoint improves.
        if t_iou < iou:
```

```
            print("Better epoch: {}\n".format(epoch))

            dict_to_save = {"seg_model": seg_model.state_dict(),
                            "bert_model": bert_model.state_dict(),
                            "optimizer": optimizer.state_dict(),
                            "epoch": epoch,
                            "args": args}

            if not args.linear_lr:
                dict_to_save["lr_scheduler"] = lr_scheduler.state_dict()

            utils.save_on_master(dict_to_save, args.output_dir +
                                 "model_best_{}.pth".format(args.model_id))

            t_iou = iou


if __name__ == "__main__":
    from args import get_parser
    parser = get_parser()
    main(parser.parse_args())
```

To access the datasets used during this work, it is necessary to use the following file as Application Programming Interface (API).

../Code/refer.py

```
#!/usr/bin/env python


"""Interface for accessing the Microsoft Refer ann_dataset.

This interface provides access to four datasets:
1) refclef
2) refcoco
3) refcoco+
4) refcocog
split by unc and google

The following API functions are defined:
Refer       - Refer api class
get_ref_ids - get ref ids that satisfy given filter conditions.
getAnnIds   - get ann ids that satisfy given filter conditions.
get_img_ids - get image ids that satisfy given filter conditions.
get_cat_ids - get category ids that satisfy given filter conditions.
loadRefs    - load refs with the specified ref ids.
loadAnns    - load anns with the specified ann ids.
loadImgs    - load images with the specified image ids.
loadCats    - load category names with the specified category ids.
getRefBox   - get ref"s bounding box [x, y, w, h] given the ref_id
showRef     - show image, segmentation or box of referred object with ref
get_mask    - get mask and area of the referred object given ref
showMask    - show mask of the referred object given ref
"""

import json
```

```python
import pickle
import time
import matplotlib.pyplot as plt
from matplotlib.collections import PatchCollection
from matplotlib.patches import Polygon
import numpy as np
from pycocotools import mask as mask_utils
from collections import defaultdict


def _is_array_like(obj):
    return hasattr(obj, "__iter__") and hasattr(obj, "__len__")


class Refer:
    """This is the Refer class.
    """

    def __init__(self, ann_file, ref_file):
        """Init Refer class.

        Provide data_root folder which contains refclef, refcoco, refcoco+ and
        refcocog also provide ann_dataset name and splitBy information e.g.,
        ann_dataset = "refcoco", splitBy = "unc"
        """

        print("Loading annotations into memory...")
        tic = time.time()
        ann_dataset = json.load(open(ann_file, "r"))
        print("Done (t={:0.2f}s)".format(time.time() - tic))
        self.ann_dataset = ann_dataset

        print("Loading referring expressions into memory...")
        tic = time.time()
        ref_dataset = pickle.load(open(ref_file, "rb"))
        print("Done (t={:0.2f}s)".format(time.time() - tic))
        self.ref_dataset = ref_dataset

        self.create_index()

    def create_index(self):
        """asdf

        create sets of mapping
        1)  refs:         {ref_id: ref}
        2)  anns:         {ann_id: ann}
        3)  imgs:         {image_id: image}
        4)  cats:         {category_id: category_name}
        5)  sents:        {sent_id: sent}
        6)  img_to_refs:    {image_id: refs}
        7)  img_to_anns:    {image_id: anns}
        8)  ref_to_ann:     {ref_id: ann}
        9)  ann_to_ref:     {ann_id: ref}
        10) cat_to_refs:    {category_id: refs}
        11) sentToRef:    {sent_id: ref}
        12) sent_to_tokens: {sent_id: tokens}
        """

```

```
        print("Creating index...")
        tic = time.time()

        # Fetch info from annotation dataset.
        anns, imgs, cats = {}, {}, {}
        cat_to_imgs, img_to_anns = defaultdict(list), defaultdict(list)
        for ann in self.ann_dataset["annotations"]:
            anns[ann["id"]] = ann
            img_to_anns[ann["image_id"]].append(ann["id"])
            cat_to_imgs[ann["category_id"]].append(ann["image_id"])
        for img in self.ann_dataset["images"]:
            imgs[img["id"]] = img
        for cat in self.ann_dataset["categories"]:
            cats[cat["id"]] = cat["name"]

        # Fetch info from referring dataset.
        refs, sents = {}, {}
        ann_to_ref, ref_to_ann = {}, {}
        ref_to_sents = defaultdict(list)
        sent_to_tokens, sent_to_ref = {}, {}
        for ref in self.ref_dataset:
            refs[ref["ref_id"]] = ref
            ann_to_ref[ref["ann_id"]] = ref["ref_id"]
            ref_to_ann[ref["ref_id"]] = ref["ann_id"]
            for sent in ref["sentences"]:
                sents[sent["sent_id"]] = sent
                sent_to_tokens[sent["sent_id"]] = sent["tokens"]
                ref_to_sents[ref["ref_id"]].append(sent["sent_id"])
                sent_to_ref[sent["sent_id"]] = ref["ref_id"]

        print("Done (t={:0.2f}s)".format(time.time() - tic))

        # Set attributes.
        self.cats = cats
        self.imgs = imgs
        self.anns = anns
        self.refs = refs
        self.sents = sents
        self.cat_to_imgs = cat_to_imgs
        self.img_to_anns = img_to_anns
        self.ann_to_ref = ann_to_ref
        self.ref_to_ann = ref_to_ann
        self.ref_to_sents = ref_to_sents
        self.sent_to_ref = sent_to_ref
        self.sent_to_tokens = sent_to_tokens

    def ann_info(self):
        """Prints information about the annotation file."""

        for key, value in self.ann_dataset["info"].items():
            print("{}: {}".format(key, value))

    def get_sent_ids(self,
                     cat_names=None,
                     cat_ids=None,
                     sup_names=None,
                     img_ids=None,
```

```
                 area_range=None,
                 is_crowd=None,
                 ann_ids=None,
                 split=None,
                 ref_ids=None,
                 sent_ids=None):
    """Get ann ids that satisfy given filter conditions.

    Args:
      cat_names:
        A list of strings specifying cat names or None if filter is
        deactivated. A single string will also work.
      cat_ids:
        A list of integers specifying cat ids or None if filter is
        deactivated. A single integer will also work.
      sup_names:
        A list of strings specifying supercategory names or None if filter
        is deactivated. A single string will also work.
      img_ids:
        A list of integers specifying cat ids or None if filter is
        deactivated. A single integer will also work.
      area_range:
        A list of two integers specifying area range (e.g. [0 inf]) or None
        if filter is deactivated.
      is_crowd:
        A boolean specifying crowd label or None if filter is deactivated.
      ann_ids:
        A list of integers specifying ann ids or None if filter is
        deactivated. A single integer will also work.
      split:
        A string specifying split label (train/val/test) or None if filter
        is deactivated.
      ref_ids:
        A list of integers specifying ann ids or None if filter is
        deactivated. A single integer will also work.
      sent_ids:
        A list of integers specifying ann ids or None if filter is
        deactivated. A single integer will also work.

    Returns:
      A list of integers specifying the sent ids.
    """

    ref_ids = self.get_ref_ids(cat_names=cat_names,
                               cat_ids=cat_ids,
                               sup_names=sup_names,
                               img_ids=img_ids,
                               area_range=area_range,
                               is_crowd=is_crowd,
                               ann_ids=ann_ids,
                               split=split,
                               ref_ids=ref_ids)

    ids = []
    for ref_id in ref_ids:
        ids += self.ref_to_sents[ref_id]
    if sent_ids is not None:
        sent_ids = sent_ids if _is_array_like(sent_ids) else [sent_ids]
```

```
            ids = [id_ for id_ in ids if id_ in sent_ids]
        return ids

    def get_ref_ids(self,
                    cat_names=None,
                    cat_ids=None,
                    sup_names=None,
                    img_ids=None,
                    area_range=None,
                    is_crowd=None,
                    ann_ids=None,
                    split=None,
                    ref_ids=None):
        """Get ann ids that satisfy given filter conditions.

        Args:
          cat_names:
            A list of strings specifying cat names or None if filter is
            deactivated. A single string will also work.
          cat_ids:
            A list of integers specifying cat ids or None if filter is
            deactivated. A single integer will also work.
          sup_names:
            A list of strings specifying supercategory names or None if filter
            is deactivated. A single string will also work.
          img_ids:
            A list of integers specifying cat ids or None if filter is
            deactivated. A single integer will also work.
          area_range:
            A list of two integers specifying area range (e.g. [0 inf]) or None
            if filter is deactivated.
          is_crowd:
            A boolean specifying crowd label or None if filter is deactivated.
          ann_ids:
            A list of integers specifying ann ids or None if filter is
            deactivated. A single integer will also work.
          split:
            A string specifying split label (train/val/test) or None if filter
            is deactivated.
          ref_ids:
            A list of integers specifying ann ids or None if filter is
            deactivated. A single integer will also work.

        Returns:
          A list of integers specifying the ref ids.
        """

        ann_ids = self.get_ann_ids(cat_names=cat_names,
                                   cat_ids=cat_ids,
                                   sup_names=sup_names,
                                   img_ids=img_ids,
                                   area_range=area_range,
                                   is_crowd=is_crowd,
                                   ann_ids=ann_ids)

        refs = [self.refs[self.ann_to_ref[ann_id]] for ann_id in ann_ids
                if ann_id in self.ann_to_ref]
```

```
        if split is not None:
            refs = [ref for ref in refs if ref["split"] == split]
        if ref_ids is not None:
            ref_ids = ref_ids if _is_array_like(ref_ids) else [ref_ids]
            refs = [ref for ref in refs if ref["id"] in ref_ids]

        ids = [ref["ref_id"] for ref in refs]
        return ids

    def get_ann_ids(self,
                    cat_names=None,
                    cat_ids=None,
                    sup_names=None,
                    img_ids=None,
                    area_range=None,
                    is_crowd=None,
                    ann_ids=None):
        """Get ann ids that satisfy given filter conditions.

        Args:
          cat_names:
            A list of strings specifying cat names or None if filter is
            deactivated. A single string will also work.
          cat_ids:
            A list of integers specifying cat ids or None if filter is
            deactivated. A single integer will also work.
          sup_names:
            A list of strings specifying supercategory names or None if filter
            is deactivated. A single string will also work.
          img_ids:
            A list of integers specifying cat ids or None if filter is
            deactivated. A single integer will also work.
          are_range:
            A list of two integers specifying area range (e.g. [0 inf]) or None
            if filter is deactivated.
          is_crowd:
            A boolean specifying crowd label or None if filter is deactivated.
          ann_ids:
            A list of integers specifying ann ids or None if filter is
            deactivated. A single integer will also work.

        Returns:
          A list of integers specifying the ann ids.
        """

        img_ids = self.get_img_ids(cat_names=cat_names,
                                   cat_ids=cat_ids,
                                   sup_names=sup_names,
                                   img_ids=img_ids)
        ann_ids_ = []
        for img_id in img_ids:
            ann_ids_ += self.img_to_anns[img_id]
        if area_range is not None:
            ann_ids_ = [ann_id for ann_id in ann_ids_
                        if self.anns[ann_id]["area"] > area_range[0]
                        and self.anns[ann_id]["area"] < area_range[1]]
        if is_crowd is not None:
            ann_ids_ = [ann_id for ann_id in ann_ids_
```

```
                       if self.anns[ann_id]["iscrowd"] == is_crowd]
        if ann_ids is not None:
            ann_ids = ann_ids if _is_array_like(ann_ids) else [ann_ids]
            ann_ids_ = [ann_id for ann_id in ann_ids_ if ann_id in ann_ids]

        return ann_ids_

    def get_img_ids(self,
                    cat_names=None,
                    cat_ids=None,
                    sup_names=None,
                    img_ids=None):
        """Get img ids that satisfy given filter conditions.

        Args:
          cat_names:
            A list of strings specifying cat names or None if filter is
            deactivated. A single string will also work.
          cat_ids:
            A list of integers specifying cat ids or None if filter is
            deactivated. A single integer will also work.
          sup_names:
            A list of strings specifying supercategory names or None if filter
            is deactivated. A single string will also work.
          img_ids:
            A list of integers specifying cat ids or None if filter is
            deactivated. A single integer will also work.

        Returns:
          A list of integers specifying the img ids.
        """

        cat_ids = self.get_cat_ids(cat_names=cat_names,
                                   sup_names=sup_names,
                                   cat_ids=cat_ids)
        ids = []
        for cat_id in cat_ids:
            ids += self.cat_to_imgs[cat_id]
        if img_ids is not None:
            img_ids = img_ids if _is_array_like(img_ids) else [img_ids]
            ids = [id_ for id_ in ids if id_ in img_ids]

        return list(set(ids))

    def get_cat_ids(self, cat_names=None, sup_names=None, cat_ids=None):
        """Get cat ids that satisfy given filter conditions.

        Args:
          cat_names:
            A list of strings specifying cat names or None if filter is
            deactivated. A single string will also work.
          sup_names:
            A list of strings specifying supercategory names or None if filter
            is deactivated. A single string will also work.
          cat_ids:
            A list of integers specifying cat ids or None if filter is
            deactivated. A single integer will also work.

```

```
        Returns:
          A list of integers specifying the cat ids.
        """

        cats = self.ann_dataset["categories"]

        if cat_names is not None:
            cat_names = cat_names if _is_array_like(cat_names) else [cat_names]
            cats = [cat for cat in cats if cat["name"] in cat_names]
        if sup_names is not None:
            sup_names = sup_names if _is_array_like(sup_names) else [sup_names]
            cats = [cat for cat in cats if cat["supercategory"] in sup_names]
        if cat_ids is not None:
            cat_ids = cat_ids if _is_array_like(cat_ids) else [cat_ids]
            cats = [cat for cat in cats if cat["id"] in cat_ids]

        ids = [cat["id"] for cat in cats]
        return ids

    def load_sents(self, ids=None):
        """Load sents with the specified ids.

        Args:
          ids:
            A list of integers specifying the sent ids or None to load all
            sents. A single integer will also work.

        Returns:
          A list of sents for all the specfied ids, or a single sent if
          ids is a single integer.
        """

        if _is_array_like(ids):
            return [self.sents[id_] for id_ in ids]
        return self.sents[ids]

    def load_refs(self, ids=None):
        """Load refs with the specified ids.

        Args:
          ids:
            A list of integers specifying the sent ids or None to load all
            refs. A single integer will also work.

        Returns:
          A list of refs for all the specfied ids, or a single sent if ids is a
          single integer.
        """

        if _is_array_like(ids):
            return [self.refs[id_] for id_ in ids]
        return self.refs[ids]

    def load_anns(self, ids=None):
        """Load anns with the specified ids.

        Args:
          ids:
```

```
                A list of integers specifying the sent ids or None to load all
                anns. A single integer will also work.

            Returns:
              A list of anns for all the specfied ids, or a single sent if ids is a
              single integer.
            """

            if _is_array_like(ids):
                return [self.anns[id_] for id_ in ids]
            return self.anns[ids]

        def load_imgs(self, ids=None):
            """Load imgs with the specified ids.

            Args:
              ids:
                A list of integers specifying the sent ids or None to load all
                imgs. A single integer will also work.

            Returns:
              A list of imgs for all the specfied ids, or a single sent if ids is a
              single integer.
            """

            if _is_array_like(ids):
                return [self.imgs[id_] for id_ in ids]
            return self.imgs[ids]

        def load_cats(self, ids=None):
            """Load cats with the specified ids.

            Args:
              ids:
                A list of integers specifying the sent ids or None to load all
                cats. A single integer will also work.

            Returns:
              A list of cats for all the specfied ids, or a single sent if ids is a
              single integer.
            """

            if _is_array_like(ids):
                return [self.cats[id_] for id_ in ids]
            return self.cats[ids]

        def show_anns(self, anns, draw_bbox=False):
            """Display the specified annotations.

            Args:
              anns:
                List of ann to display. It will also work with a single ann.
              draw_bbow:
                A boolean specifying if the bounding box should be drawn.
            """

            anns = anns if _is_array_like(anns) else [anns]

```

```
        if "segmentation" in anns[0] or "keypoints" in anns[0]:
            dataset_type = "instances"
        elif "caption" in anns[0]:
            dataset_type = "captions"
        else:
            raise Exception("dataset_type not supported")
        if dataset_type == "instances":
            ax = plt.gca()
            ax.set_autoscale_on(False)
            polygons = []
            color = []
            for ann in anns:
                c = (np.random.random((1, 3))*0.6 + 0.4).tolist()[0]
                if "segmentation" in ann:
                    if isinstance(ann["segmentation"], list):
                        # polygon
                        for seg in ann["segmentation"]:
                            poly = np.array(seg).reshape((int(len(seg)/2), 2))
                            polygons.append(Polygon(poly))
                            color.append(c)
                    else:
                        # mask
                        t = self.imgs[ann["image_id"]]
                        if isinstance(ann["segmentation"]["counts"], list):
                            rle = mask_utils.frPyObjects([ann["segmentation"]],
                                                         t["height"],
                                                         t["width"])
                        else:
                            rle = [ann["segmentation"]]
                        m = mask_utils.decode(rle)
                        img = np.ones((m.shape[0], m.shape[1], 3))
                        if ann["iscrowd"] == 1:
                            color_mask = np.array([2.0, 166.0, 101.0])/255
                        if ann["iscrowd"] == 0:
                            color_mask = np.random.random((1, 3)).tolist()[0]
                        for i in range(3):
                            img[:, :, i] = color_mask[i]
                        ax.imshow(np.dstack((img, m*0.5)))
                if "keypoints" in ann and isinstance(ann["keypoints"], list):
                    # turn skeleton into zero-based index
                    sks = np.array(
                        self.loadCats(ann["category_id"])[0]["skeleton"]
                    ) - 1
                    kp = np.array(ann["keypoints"])
                    x = kp[0::3]
                    y = kp[1::3]
                    v = kp[2::3]
                    for sk in sks:
                        if np.all(v[sk] > 0):
                            plt.plot(x[sk], y[sk], linewidth=3, color=c)
                    plt.plot(x[v > 0], y[v > 0], "o", markersize=8,
                             markerfacecolor=c, markeredgecolor="k",
                             markeredgewidth=2)
                    plt.plot(x[v > 1], y[v > 1], "o", markersize=8,
                             markerfacecolor=c, markeredgecolor=c,
                             markeredgewidth=2)

                if draw_bbox:
```

```
                    [bbox_x, bbox_y, bbox_w, bbox_h] = ann["bbox"]
                    poly = [[bbox_x, bbox_y], [bbox_x, bbox_y + bbox_h],
                            [bbox_x + bbox_w, bbox_y + bbox_h],
                            [bbox_x + bbox_w, bbox_y]]
                    np_poly = np.array(poly).reshape((4, 2))
                    polygons.append(Polygon(np_poly))
                    color.append(c)

            p = PatchCollection(polygons, facecolor=color, linewidths=0,
                                alpha=0.4)
            ax.add_collection(p)
            p = PatchCollection(polygons, facecolor="none", edgecolors=color,
                                linewidths=2)
            ax.add_collection(p)
        elif dataset_type == "captions":
            for ann in anns:
                print(ann["caption"])

    def ann_to_RLE(self, ann):
        """Convert annotation to RLE.

        Convert annotation which can be polygons, uncompressed RLE to RLE.
        :return: binary mask (numpy 2D array)

        Args:
          ann:
            Annotation object.

        Returns:
          A numpy 2D array specifying the binary mask.
        """

        t = self.imgs[ann["image_id"]]
        h, w = t["height"], t["width"]
        segm = ann["segmentation"]
        if isinstance(segm, list):
            # polygon -- a single object might consist of multiple parts
            # we merge all parts into one mask rle code
            rles = mask_utils.frPyObjects(segm, h, w)
            rle = mask_utils.merge(rles)
        elif isinstance(segm["counts"], list):
            # uncompressed RLE
            rle = mask_utils.frPyObjects(segm, h, w)
        else:
            # rle
            rle = ann["segmentation"]
        return rle

    def ann_to_mask(self, ann):
        """Convert annotation to binary mask.

        Convert annotation which can be polygons, uncompressed RLE, or RLE to
        binary mask_utils.

        Args:
          ann:
            Annotation object.

```

```
        Returns:
          A numpy 2D array specifying the binary mask.
        """

        rle = self.ann_to_RLE(ann)
        m = mask_utils.decode(rle)
        return m
```

For the use of the model, it has been useful to create a file in Python containing the same model as an object. Attached below.

../Code/model.py

```python
import torch


class Model(torch.nn.Module):
    def __init__(self, seg_model, bert_model):
        super().__init__()
        self.seg_model = seg_model
        self.bert_model = bert_model

    def forward(self, sent, attention, img):
        last_hidden_state = self.bert_model(sent,
                                            attention_mask=attention)[0]
        embedding = last_hidden_state[:, 0, :]
        outputs, _, _ = self.seg_model(img, embedding.squeeze(1))

        outputs = outputs["out"]

        return outputs

    def eval(self):
        self.seg_model.eval()
        self.bert_model.eval()

    def train(self):
        self.seg_model.train()
        self.bert_model.train()
```

B.2 Website

In relation to the website, we will show the most important files created. We will separate between the front end (see Appendix B.2.1) and the back end (see Appendix B.2.2 on page 112)

B.2.1 Front End

Within the front end the main file is obviously `index.html`, but due to its extension it has not been included. It also doesn't bring too much extra functionality to work. Yes, the stylesheet CSS is included below.

../Website/css/main.css

```css
html {
    scroll-behavior: smooth;
}

section {
    padding-bottom: 2em;
}

#gallery {
    width: 100%;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    justify-content: flex-start;
    margin: 25px -5px 25px -5px;
    max-height: 500px;
    overflow-y: auto;
}

#gallery > img {
    width: 175px;
    height: 175px;
    max-width: 100%;
    margin: 5px;
    object-fit: cover;
    object-position: center;
}

#gallery > img:hover {
    cursor: pointer;
    border: thick solid black;
}

#img-selected {
    width: 350px;
    max-width: 100%;
    margin: auto;
    display: none;
    padding-bottom: 2em;
}

#re-selected {
    display: none;
}

#img-segmented {
    display: block;
    width: 575px;
    max-width: 100%;
    margin: auto;
}
```

In addition, as a fundamental part of the interactivity of the web, the file containing the code of JS is fundamental, which makes the requests to the API of the back end to collect the information.

../Website/js/main.js

```
// Show selected image (from gallery, url or local storage).
function showSelectedImg(src, method) {
    let imgSelected = document.getElementById("img-selected");
    imgSelected.setAttribute("src", src);
    imgSelected.setAttribute("data-method", method);
    imgSelected.style.display = "block";
    let imgSelectedWarn = document.getElementById("img-selected-warn");
    imgSelectedWarn.style.display = "none";
    // Scroll to results section.
    let resultsSection = document.getElementById("sec:results")
    resultsSection.scrollIntoView({ behavior: "smooth" });
}


// Select image from website gallery.
function selectImg(event) {
    let selectedImgSrc = event.target.src;
    showSelectedImg(selectedImgSrc, "gallery");
}


// Add image via URL.
function addImg() {
    let imgUrl = document.getElementById("img-url").value;
    showSelectedImg(imgUrl, "url");
    return false; // Prevent form to be submitted.
}


// Upload image locally from computer.
function uploadImg() {
    let imgLocal = document.getElementById("img-local");
    let uploadedImg = imgLocal.files[0];

    const fileReader = new FileReader();
    fileReader.addEventListener("load", function () {
        showSelectedImg(this.result, "local");
    });
    fileReader.readAsDataURL(uploadedImg);

    return false; // Prevent form to be submitted.
}


// Checks if an image have been already selected.
function isImgSelected() {
    let imgSelected = document.getElementById("img-selected");
    if (imgSelected.src === "")
        return false;
    return true;
}


// Enter referring expression.
function addReferringExpression() {
```

```
    if (!isImgSelected()) {
        let imgSelectedWarn = document.getElementById("img-selected-warn");
        imgSelectedWarn.classList.remove("alert-warning");
        imgSelectedWarn.classList.add("alert-danger");
        imgSelectedWarn.scrollIntoView({ behavior: "smooth" });
        return false;
    }

    let referringExpression = document.getElementById("referring-expression");
    if (referringExpression.value === "")
        return false;
    let reSelected = document.getElementById("re-selected");
    reSelected.textContent = referringExpression.value;
    reSelected.style.display = "block";
    let reSelectedWarn = document.getElementById("re-selected-warn");
    reSelectedWarn.style.display = "none";

    // Execute code.
    segmentImg();

    return false; // Prevent form to be submitted.
}

function addReferringExpressionFromString(referringExpression) {
    let reContainer = document.getElementById("referring-expression");
    reContainer.value = referringExpression;
    addReferringExpression();
}



// Populate website gallery with random images from MSCOCO dataset.
function populateGallery() {
    const gallerySize = 12;
    let gallery = document.getElementById("gallery");
    gallery.innerHTML = "";
    let imgNumbers = [];
    for (let i = 0; i < gallerySize; ++i) {
        // Choose random number differnt from previous.
        let imgNumber = Math.round(Math.random()*(imgFileNames.length - 1));
        while (imgNumbers.includes(imgNumber))
            imgNumber = Math.round(Math.random()*(imgFileNames.length - 1));
        imgNumbers.push(imgNumber);

        // Set gallery image.
        let imgFileName = imgFileNames[imgNumber];
        let imgSrc = "datasets/refcoco/images/" + imgFileName;
        let galleryImg = document.createElement("img");
        galleryImg.setAttribute("src", imgSrc);
        galleryImg.setAttribute("alt", imgFileName);
        galleryImg.classList.add("rounded");
        galleryImg.setAttribute("data-toggle", "tooltip");
        galleryImg.setAttribute("title", "Select image " + imgFileName);
        galleryImg.onclick = selectImg;
        gallery.appendChild(galleryImg);
    }
}

```

```
populateGallery();


// Start audio recording.
function startAudio() {
    initAudio();
    let audioContainer = document.getElementById("audio");
    audioContainer.style.display = "block";
}

// window.addEventListener("load", startAudio);


// Stop audio recording.
function stopAudio() {
    let audioContainer = document.getElementById("audio");
    audioContainer.style.display = "none";
}


// Activate all tooltips.
$(function () {
    $('[data-toggle="tooltip"]').tooltip();
})


// Segment image with referring expression.
function segmentImg() {
    let imgSelected = document.getElementById("img-selected");
    let imgSrc = imgSelected.src;
    let reSelected = document.getElementById("re-selected");
    let referringExpression = reSelected.innerText;

    let formData = new FormData();
    formData.append("referringExpression", referringExpression);
    formData.append("imgMethod", imgSelected.dataset.method);
    formData.append("imgSrc", imgSrc);

    fetch("api/comprehend.php", {
        method: "POST",
        body: formData
    }).then(response => response.json())
        .then(response => {
            console.log(response);
            let img = document.getElementById("img-segmented");
            img.setAttribute("src", response['resultImgSrc']);
        });
}


// Toggle recording auxiliary function.
function toggleRecordingAux(event) {
    if (event.classList.contains("recording")) {
        // Start recording.
        event.title = "Stop recording";
    } else {
        // Stop recording.
```

```
        event.title = "Start recording";
        saveAudio();
        stopAudio();
    }
}


// Show warning message.
$(document).ready(function(){
        $("#warningModal").modal('show');
});
```

B.2.2 Back End

In the back end highlighting two files, which are the ones that really are API. The first is the one that deals with actually performing the main task of this work, that is, segmentation.

../Website/api/comprehend.php

```php
<?php
header('Content-Type:application/json');

$baseFileName = 'results/' . uniqid();

switch ($_POST['imgMethod']) {
    case 'gallery':
        $fileName = $baseFileName . '.jpg';
        copy($_POST['imgSrc'], $fileName);
        break;
    case 'url':
        $path = parse_url($_POST['imgSrc'], PHP_URL_PATH);
        $extension = pathinfo($path, PATHINFO_EXTENSION);
        $fileName = $baseFileName . '.' . $extension;
        file_put_contents($fileName, file_get_contents($_POST['imgSrc']));
        break;
    case 'local':
        $extension = explode('/', mime_content_type($_POST['imgSrc']))[1];
        $fileName = $baseFileName . '.' . $extension;
        file_put_contents($fileName, base64_decode(
            explode(';base64,', $_POST['imgSrc'])[1]
        ));
        break;
}

$referringExpression = $_POST['referringExpression'];

$command = '. Code/.venv/bin/activate 2>&1 &&'.
         ' XDG_CACHE_HOME=.cache/ MPLCONFIGDIR=.cache/' .
         ' python Code/comprehend.py' .
         ' --resume Code/checkpoints/model_refcoco.pth' .
         ' --img ' . $fileName .
         ' --sent "' . $referringExpression . '"' .
         ' --device cpu' .
         ' --output ' . $baseFileName . '.out.jpg 2>&1';

```

```

exec($command, $outputCommand);

$output = [
    'command' => $command,
    'outputCommand' => $outputCommand,
    'resultImgSrc' => 'api/' . $baseFileName . '.out.jpg'
];

if (isset($_POST['debug'])) {
    print_r($_POST);
    echo $command;
    echo $outputCommand;
    print_r($outputCommand);
    // var_dump($outputCommand);
}
else
    echo json_encode($output);
?>
```

The following file constitutes the part of the API of the back end is the one for converting audio to text, which is shown below.

../Website/api/uploadWav.php

```
<?php
$fileName = 'audio/' . uniqid() . '.wav';

move_uploaded_file($_FILES["audio"]["tmp_name"], $fileName);

$command = '. Code/.venv/bin/activate 2>&1 &&'.
           ' XDG_CACHE_HOME=.cache/ TMP=.cache/' .
           ' python -W ignore Code/Prueba/main.py' .
           ' --file ' . $fileName . ' 2>&1';

exec($command, $outputCommand);

$output = [
    'command' => $command,
    'outputCommand' => $outputCommand,
];

if (isset($_POST['debug'])) {
    var_dump($output);
    print_r($outputCommand);
}
else
    echo json_encode($output);


?>
```

In addition, the Python files that are executed in the back end after being called by the different functions of API are added below. These are, `comprehend.py` and `silero.py`.

../Code/comprehend.py

```python
"""File for the comprehension (forward of the model).

A more detailed explanation.
"""

import torch
from transformers import BertModel
from lib import segmentation
from model import Model
import torchvision.transforms.transforms as T
from transformers import BertTokenizer
import PIL

import utils

import time


def main(args):
    tic = time.time()

    # Segmentation model.
    seg_model = segmentation.deeplabv3_resnet101(num_classes=2,
                                                 aux_loss=False,
                                                 pretrained=False,
                                                 args=args)

    print("hey from here: ", time.time() - tic)
    # BERT model.
    bert_model = BertModel.from_pretrained(args.ck_bert)



    # Load checkpoint.
    device = torch.device(args.device)
    ticAux = time.time()
    checkpoint = torch.load(args.resume, map_location=device)
    print("extra time", time.time() - ticAux)

    bert_model.load_state_dict(checkpoint["bert_model"], strict=False)
    seg_model.load_state_dict(checkpoint["model"], strict=False)

    # Define model and sent to device.
    model = Model(seg_model, bert_model)

    model.to(device)

    model.eval()

    print("loading of model time: ", time.time() - tic)
    tic = time.time()

    img_raw = PIL.Image.open(args.img)


    max_tokens = 20
```

```
    attention_mask = [0] * max_tokens
    padded_input_ids = [0] * max_tokens

    tokenizer = BertTokenizer.from_pretrained(args.bert_tokenizer)
    input_ids = tokenizer.encode(text=args.sent,
                                  add_special_tokens=True)

    # truncation of tokens
    input_ids = input_ids[:max_tokens]

    padded_input_ids[:len(input_ids)] = input_ids
    attention_mask[:len(input_ids)] = [1]*len(input_ids)

    sents = torch.tensor(padded_input_ids).unsqueeze(0)
    attention_mask = torch.tensor(attention_mask).unsqueeze(0)

    transforms = T.Compose([
        T.ToTensor(),
        T.Normalize(mean=[0.485, 0.456, 0.406],
                    std=[0.229, 0.224, 0.225])
    ])

    img = transforms(img_raw)
    imgs = img.unsqueeze(0)

    imgs, attentions, sents = \
        imgs.to(device), attention_mask.to(device), sents.to(device)

    print("prepare inputs: ", time.time() - tic)
    tic = time.time()


    with torch.no_grad():
        outputs = model(sents, attentions, imgs)
        masks = outputs.argmax(1)

    mask = masks.squeeze(0).cpu()

    print("forward model with no_grad: ", time.time() - tic)
    tic = time.time()


    utils.save_figure(img_raw, args.sent, mask, args.output)

    print("savefigure: ", time.time() - tic)
    tic = time.time()


if __name__ == "__main__":
    from args import get_parser
    parser = get_parser()
    main(parser.parse_args())
```

../Code/silero.py

```
"""Code for the Speech to Text (STT) task.

Using Silero model.
"""

import argparse
import torch
import zipfile
import torchaudio
from glob import glob


device = torch.device('cpu')

model, decoder, utils = torch.hub.load(repo_or_dir='snakers4/silero-models',
                                       model='silero_stt',
                                       language='en',
                                       device=device,
                                       verbose=False)

(read_batch, split_into_batches,
 read_audio, prepare_model_input) = utils

parser = argparse.ArgumentParser(description="ArgumentParser")
parser.add_argument("--file", help="Name of audio file", required=True)

args = parser.parse_args()

test_files = glob(args.file)
batches = split_into_batches(test_files, batch_size=10)
input = prepare_model_input(read_batch(batches[0]),
                            device=device)

output = model(input)
for example in output:
    print(decoder(example.cpu()))
```

B.3 Server

For the connection and use of the server, the main files that have been necessary are shown below. The first one is the script used to synchronize files between the local computer and the remote servers, attached below.

../Utils/newServer

```
#!/bin/bash

# Sync all files with remote server (excluding GÇIT, datasets, flycheck, Python
# venv and other caches and datasets).

rsync -e "ssh -p 6969" \
      -avzhP \
```

```
      --delete \
      --exclude checkpoints/ \
      --exclude .git \
      --exclude __pycache__ \
      --exclude '.#*' \
      --exclude 'flycheck_*' \
      --exclude .gitignore \
      --exclude .venv \
      --exclude .cache \
      --exclude results \
      --exclude audio \
      --exclude images \
      /home/david/Documents/UPC/Cuatrimestre\ 9/Bachelor\'s\ Thesis/ \
      root@recomprehension.com:/var/www/html/thesis/
```

In addition, the server makes use of a management system called Slurm, which is an open-source job scheduler so that it is possible to make use of the computational resources of multiple servers by numerous users and do all this in an orderly manner. For this, this program has a very specific syntax with which to execute the desired code. A typical script for this type of task is shown below.

../Utils/launch

```
#!/bin/bash


#SBATCH --job-name=dvd-test      # create a short name for your job
#SBATCH --nodes=1                # node count
#SBATCH --ntasks=1               # total number of tasks across all nodes
#SBATCH --cpus-per-task=1        # cpu-cores per task
#SBATCH --mem=4G                 # total memory per node
#SBATCH --gres=gpu:1             # number of gpus per node
#SBATCH --partition=gpu          # partition requested
#SBATCH --time=08:00:00          # total run time limit (HH:MM:SS)
#SBATCH --output=out.txt         # file for script's standard output
#SBATCH --mail-type=begin        # send mail when job begins
#SBATCH --mail-type=end          # send mail when job ends
#SBATCH --mail-type=fail         # send mail if job fails
#SBATCH --mail-user=david.alvarez.rosa@yandex.com


source .venv/bin/activate
python train.py --dataset refcoco \
       --model_id model \
       --image_root /scratch/gobi1/datasets/MSCOCO/images/train2014/ \
       --pretrained \
       --workers 4 \
       --batch_size 16
```

Write what should not be forgotten.
—Isabel ALLENDE

Appendix C

Supplementary Material

THIS APPENDIX will include all the other extra auxiliary material that has not been considered important enough to include it as a main part of the document. That is, everything that wants to be remembered but is not considered important enough to be part of the bulk of the thesis will be included here.

C.1 Activation Functions

The activation functions as already discussed in Chapter 2 on page 9 are fundamental for the creation of Artificial Neural Network (ANN). This is mainly due to the need to introduce non-linearities to the models, so as to facilitate the adjustment of these to complex functions: the nature is highly non-linear and it would be impossible to achieve useful results using only linear functions for the adjustment of data. In this section, three of the most used activation functions will be discussed and compared: Rectified Linear Unit (ReLU), the hyperbolic tangent and the sigmoid function.

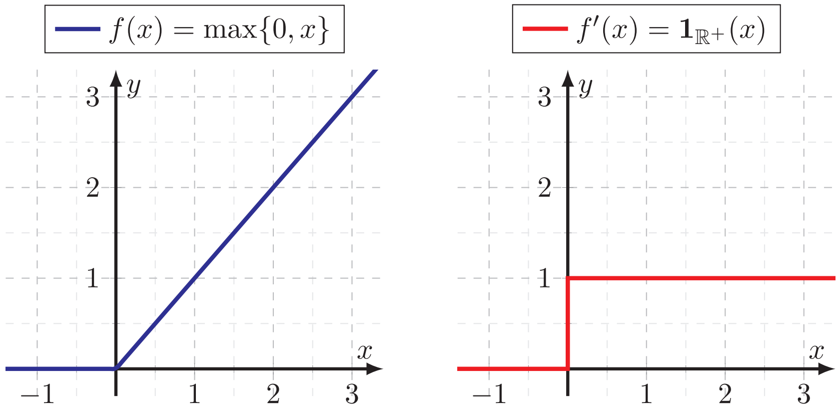
ReLU is one of the simplest, known and most widely used activation functions. It is the function that is defined by the following expression,

$$f(x) = \max\{0, x\}, \tag{C.1}$$

with derivative $f'(x) = \mathbf{1}_{\mathbb{R}^+}(x)$. It presents several advantages such as: sparse activation (if the neuron values were random, only 50% of the neurons would have non-zero activation), efficient gradient propagation (it does not present problems of vanishing gradient or—at least—it presents fewer problems than the activation functions that saturate in both directions) and the computation of the activation is very efficient at the computational level. In Figure C.1 on the next page both the graph of the function and its derivative are shown.

It should be noted that in some applications complications may occur with the use of the activation function ReLU. This is mainly due to three factors: it is not differentiable in 0, it is not zero-centered (which would be a desirable feature in some cases) and it is not a bounded function, which could lead to overflow problems at the computational level.

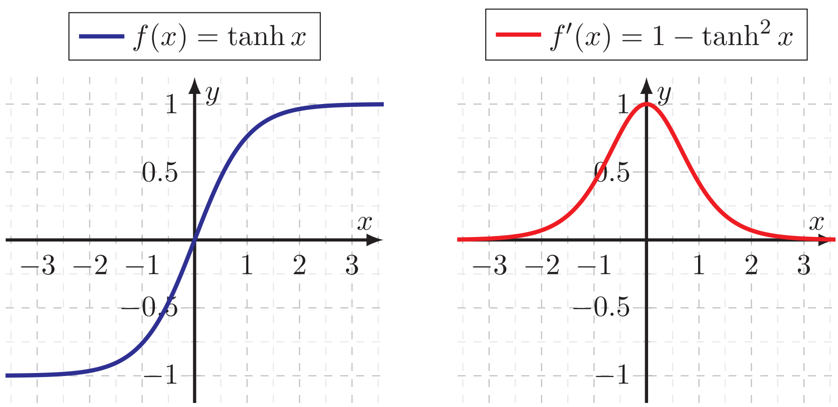
Another of the activation functions typically used in this area is that of *hyperbolic tangent*. This function, whose graph and derivative are represented in Figure C.2,



(a) Activation function **(b)** Activation function derivative

Figure C.1. Rectified Linear Unit (ReLU) activation function and derivative. Figures create by the author (both).

presents odd symmetry and saturates symmetrically. Vanishing gradient issues may appear when using this feature.



(a) Activation function **(b)** Activation function derivative

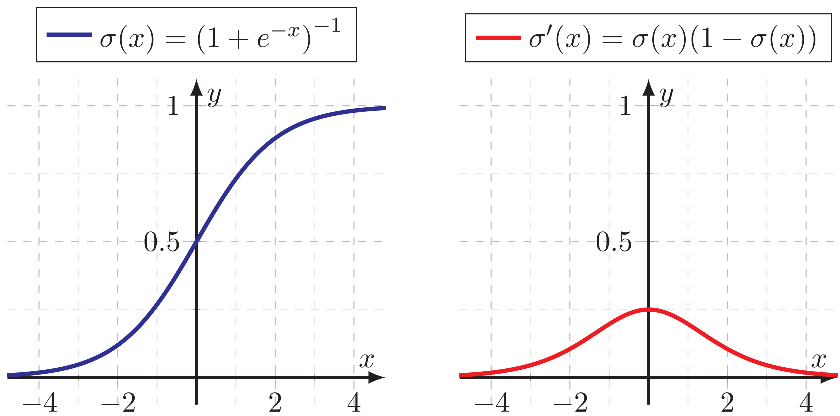
Figure C.2. Hyperbolic tangent activation function and derivative. Figures created by the author (both).

Finally, another of the known activation functions is that of the logistic or sigmoid function. This function is well known within the scope of Machine Learning (ML) for its use in the logistic regression, defined by the following expression,

$$\sigma(x) = (1 + e^{-x})^{-1}, \tag{C.2}$$

and whose derivative can be expressed in terms of the original function as $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

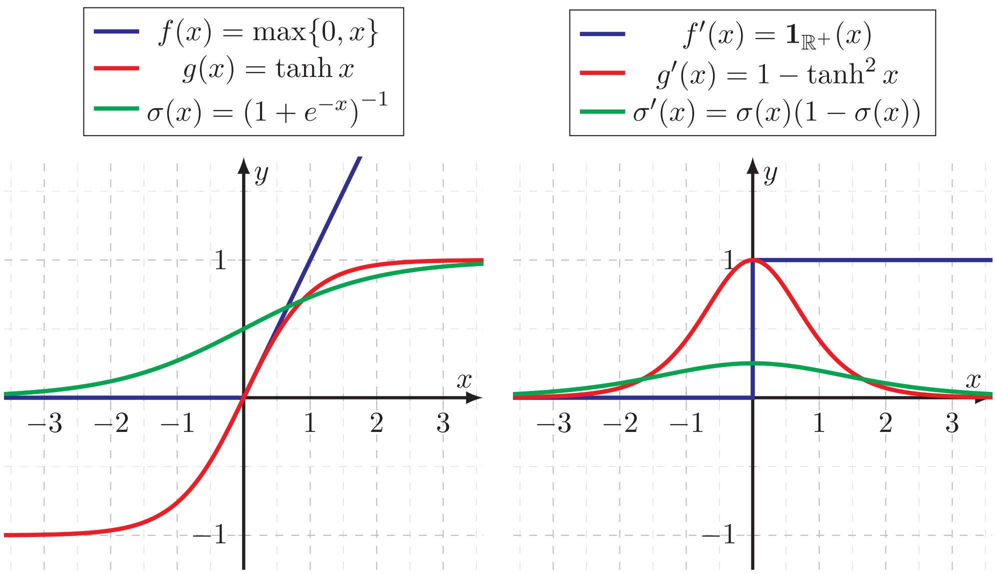
It is a function (see Figure C.3) that has good mathematical properties such as continuity and differentiability throughout its domain and that limits the activation of the neuron to the range $[0, 1]$. This function is especially useful in the case of binary classification, but it is not widely used today mainly because: it may cause the vanishing gradient problem, it is not centered on 0 and its calculation is computationally expensive.



(a) Activation function **(b)** Activation function derivative

Figure C.3. Sigmoid activation function and derivative (also called logistic function and soft step). Figures created by the author (both).

The activation functions described: ReLU, hyperbolic tangent and the sigmoid function are plotted together in Figure C.4 on the following page, where they can be compared.



(a) Activation functions

(b) Activation function derivatives

Figure C.4. Activation function comparison. Rectified Linear Unit (ReLU), hyperbolic tangent and sigmoid activation functions are plotted overlapping for a better comparison. Figures created by the author (both).

Bibliography

The complete bibliography has been divided into three: primary sources (which contains all the citations that appear in the text), figure sources and quotation sources.

Primary Sources

Ali21 Alibaba Group Holding Limited. *China Arduino Robot Arm*. https://www.alibaba.com/countrysearch/CN/arduino-robot-arm.html. [Online; accesed 16 March 2021]. Mar. 2021.

Asi21 Asia Times Online. *China's robot market is still No. 1*. https://asiatimes.com/2019/09/chinas-robot-market-still-no-1/. [Online; accesed 16 March 2021]. Mar. 2021.

Bel+20 Miriam Bellver, Carles Ventura, Carina Silberer, Ioannis Kazakos, Jordi Torres, and Xavier Giró-i-Nieto. "RefVOS: A Closer Look at Referring Expressions for Video Object Segmentation". In: *CoRR* abs/2010.00263 (2020). arXiv: 2010.00263. url: https://arxiv.org/abs/2010.00263.

Ber21 Tim Berners-Lee. *Introduction to Web Accessibility — Accessibility in Context*. https://www.w3.org/WAI/fundamentals/accessibility-intro/. [Online; accessed 6 April of 2021]. 2021.

Bis+95 C.M. Bishop, P.N.C.C.M. Bishop, G. Hinton, and Oxford University Press. *Neural Networks for Pattern Recognition*. Advanced Texts in Econometrics. Clarendon Press, 1995. isbn: 9780198538646.

Cau87 Maureen Caudill. "Neural Networks Primer, Part I". In: *AI Expert* 2.12 (Dec. 1987), pp. 46–52. issn: 0888-3785.

Che+17 Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv* (2017). arXiv: 1706.05587 [cs.CV].

Che+19a Ding-Jie Chen, Songhao Jia, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. "See-Through-Text Grouping for Referring Image Segmentation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019, pp. 7453–7462. doi: 10.1109/ICCV.2019.00755.

Che+19b Yi-Wen Chen, Yi-Hsuan Tsai, Tiantian Wang, Yen-Yu Lin, and Ming-Hsuan Yang. *Referring Expression Object Segmentation with Caption-Aware Consistency*. 2019. arXiv: `1910.04748 [cs.CV]`.

Cho+14 Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv preprint arXiv:1406.1078* (2014). arXiv: `1406.1078 [cs.CL]`.

Com18 Computer Science Wiki. *Max-pooling/Pooling — Computer Science Wiki*. [Online; accessed 16 March 2021]. 2018. url: `%5Curl%7Bhttps://computersciencewiki.org/index.php?title=Max-pooling_/_Pooling&oldid=7839%7D`.

Cor+18 Marcella Cornia, Lorenzo Baraldi, Hamed R. Tavakoli, and Rita Cucchiara. "Towards Cycle-Consistent Models for Text and Image Retrieval". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

Dev+19 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1. Minneapolis, Minnesota: Association for Computational Linguistics, May 2019, pp. 4171–4186.

Eve+10 M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

Fag+18 Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. "VSE++: Improving Visual-Semantic Embeddings with Hard Negatives". In: *Proceedings of the British Machine Vision Conference (BMVC)*. July 2018. url: `https://github.com/fartashf/vsepp`.

Gan73 Henry Laurence Gantt. *Work Wages and Profits (Management in History No 41)*. Hive Publishing Company, Sept. 1973. isbn: 0879600489.

GB10 Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *AISTATS*. Ed. by Yee Whye Teh and D. Mike Titterington. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256. url: `http://dblp.uni-trier.de/db/journals/jmlr/jmlrp9.html#GlorotB10`.

Gir20 Xavier Giró-i-Nieto. *All lectures on Deep Learning at UPC ETSETB TelecomBCN*. `https://github.com/telecombcn-dl/lectures-all`. [Online; accessed 1 October 2020]. 2020.

Gov21 Government of Aragon (Spain). *CO_2 Emission Calculator*. `http://calcarbono.servicios4.aragon.es/`. [Online; accessed 6 April of 2021]. 2021.

GQ01 Ramazan Gençay and Min Qi. "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging". In: *IEEE Transactions on Neural Networks* 12.4 (2001), pp. 726–734. doi: `10.1109/72.935086`.

HBB19 Radja Hachilif, Riyadh Baghdadi, and Fatima Benhamida. "Graduation Thesis Implementing and Optimizing Neural Networks using Tiramisu". PhD thesis. Ecole Nationale Supérieure d'Informatique, June 2019.

He+15 K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. doi: `10.1109/ICCV.2015.123`.

He+16 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778. arXiv: `1512.03385 [cs.CV]`.

Hu+20 Zhiwei Hu, Guang Feng, Jiayu Sun, Lihe Zhang, and Huchuan Lu. "Bi-Directional Relationship Inferring Network for Referring Image Segmentation". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4423–4432. doi: `10.1109/CVPR42600.2020.00448`.

Hua+20 Shaofei Huang, Tianrui Hui, Si Liu, Guanbin Li, Yunchao Wei, Jizhong Han, Luoqi Liu, and Bo Li. "Referring Image Segmentation via Cross-Modal Progressive Comprehension". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 10488–10497. arXiv: `2010.00514 [cs.CV]`.

IAM21 IAM Robotics, LLC. *The Future of Material Handling.* `https://www.iamrobotics.com/`. [Online; accesed 16 March 2021]. Mar. 2021.

Kaz+14 Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L. Berg. "ReferIt Game: Referring to Objects in Photographs of Natural Scenes". In: *EMNLP*. Oct. 2014. url: `http://tamaraberg.com/referitgame/`.

KB14 Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint* (2014). eprint: `1412.6980` (cs.LG).

LeC98 Yann LeCun. *LeNet-5, Convolutional Neural Networks — Personal Website*. `http://yann.lecun.com/exdb/lenet/`. [Online; accessed 15 February of 2021]. 1998.

Li+18 Ruiyu Li, Kaican Li, Yi-Chun Kuo, Michelle Shu, Xiaojuan Qi, Xiaoyong Shen, and Jiaya Jia. "Referring Image Segmentation via Recurrent Refinement Networks". In: *CVPR*. 2018.

Lin+14 Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755. arXiv: 1405.0312 [cs.CV].

Liu+17 Chenxi Liu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, and Alan Yuille. "Recurrent Multimodal Interaction for Referring Image Segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1271–1280. arXiv: 1703.07939 [cs.CV].

Liu+19a Daqing Liu, Hanwang Zhang, Zheng-Jun Zha, and Wu Feng. "Learning to Assemble Neural Module Tree Networks for Visual Grounding". In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

Liu+19b Runtao Liu, Chenxi Liu, Yutong Bai, and Alan L Yuille. "CLEVR-Ref+: Diagnosing Visual Reasoning with Referring Expressions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4185–4194. arXiv: 1901.00850 [cs.CV].

Liu+19c Xihui Liu, Zihao Wang, Jing Shao, Xiaogang Wang, and Hongsheng Li. "Improving referring expression grounding with Cross-modal Attention-guided Erasing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1950–1959. arXiv: 1903.00839 [cs.CV].

LKX20 Fei-Fei Li, Ranjay Krishna, and Danfei Xu. *CS231n: Convolutional Neural Networks for Visual Recognition*. http://cs231n.stanford.edu/. [Online; accessed 1 October 2020]. 2020.

Lu+19 Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *arXiv preprint* (2019). eprint: 1908.02265 (cs.CV).

Mao+16 Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. "Generation and Comprehension of Unambiguous Object Descriptions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 11–20. arXiv: 1511.02283 [cs.CV].

Mar+18 Edgar Margffoy-Tuay, Juan C Pérez, Emilio Botero, and Pablo Arbeláez. "Dynamic Multimodal Instance Segmentation guided by natural language queries". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 630–645. arXiv: 1807.02257 [cs.CV].

McK18 Joe McKendrick. *Artificial Intelligence Will Replace Tasks, Not Jobs*. https://www.forbes.com/sites/joemckendrick/2018/08/14/artificial-intelligence-will-replace-tasks-not-jobs/. [Online; accesed 17 March 2021]. Aug. 2018.

Neu14 John von Neumann. *Common Mistakes in using Statistics: Spotting and Avoiding Them.* `https://web.ma.utexas.edu/users/mks/statmistakes/ovefitting.html`. [Online; accessed 20 February 2021]. 2014.

Ng20 Andrew Ng. *Machine Learning — Stanford University (via Coursera).* `https://www.coursera.org/learn/machine-learning`. [Online; accessed 1 October 2020]. 2020.

Nil09 Nils J. Nilsson. *The Quest for Artificial Intelligence.* 1st. USA: Cambridge University Press, 2009. isbn: 0521122937.

NKM20 Andrew Ng, Kian Katanforoosh, and Younes Bensouda Mourri. *Deep Learning Specialization — DeepLearning.AI (via Coursera).* `https://www.coursera.org/specializations/deep-learning`. [Online; accessed 1 October 2020]. 2020.

Poc19 Pocono Record — Daily Newspaper. *Drone used to track suspects after Rte. 22 crash News.* `https://www.poconorecord.com/news/20190311/drone-used-to-track-suspects-after-rte-22-crash`. [Online; accessed 12 February of 2021]. 2019.

QDW20 Yanyuan Qiao, Chaorui Deng, and Qi Wu. "Referring Expression Comprehension: A Survey of Methods and Datasets". In: *IEEE Transactions on Multimedia* (2020). arXiv: `2007.09554 [cs.CV]`.

Qiu+20 Shuang Qiu, Yao Zhao, Jianbo Jiao, Yunchao Wei, and Shikui Wei. "Referring Image Segmentation by Generative Adversarial Learning". In: *IEEE Transactions on Multimedia* 22.5 (2020), pp. 1333–1344. doi: `10.1109/TMM.2019.2942480`.

Rez+19 Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. "Generalized Intersection over Union: A metric and a loss for bounding box regression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* June 2019, pp. 658–666. arXiv: `1902.09630 [cs.CV]`.

RFB15 Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention.* Springer. 2015, pp. 234–241. arXiv: `1505.04597 [cs.CV]`.

Vas+17 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* Red Hook, NY, USA: Curran Associates Inc., June 2017, pp. 6000–6010.

Vey20 Alexander Veysov. "Toward's an ImageNet Moment for Speech-to-Text". In: *The Gradient* (2020).

Wan+19 Peng WANG, Qi WU, Jiewei CAO, Chunhua SHEN, Lianli GAO, and Anton van den HENGEL. "Neighbourhood Watch: Referring Expression Comprehension via Language-Guided Graph Attention Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1960–1968. arXiv: 1812.04794 [cs.CV].

Wik21a WIKIPEDIA CONTRIBUTORS. *Jaccard index — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=1009813550. [Online; accessed 9-April-2021]. 2021.

Wik21b WIKIPEDIA CONTRIBUTORS. *Long short-term memory — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1005032489. [Online; accessed 10 March of 2021]. 2021.

Wik21c WIKIPEDIA CONTRIBUTORS. *Overfitting — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=1016721642. [Online; accessed 9 April 2021]. 2021.

Yan+19 Zhengyuan YANG, Boqing GONG, Liwei WANG, Wenbing HUANG, Dong YU, and Jiebo LUO. "A Fast and Accurate One-Stage Approach to Visual Grounding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4683–4693. arXiv: 1908.06354 [cs.CV].

Ye+21 Linwei YE, Mrigank ROCHAN, Zhi LIU, Xiaoqin ZHANG, and Yang WANG. "Referring Segmentation in Images and Videos with Cross-Modal Self-Attention Network". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/tpami.2021.3054384. URL: http://dx.doi.org/10.1109/TPAMI.2021.3054384.

YLY19a Sibei YANG, Guanbin LI, and Yizhou YU. "Cross-Modal Relationship Inference for Grounding Referring Expressions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

YLY19b Sibei YANG, Guanbin LI, and Yizhou YU. "Dynamic Graph Attention for Referring Expression Comprehension". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4644–4653. arXiv: 1909.08164 [cs.CV].

Yu+18 Licheng YU, Zhe LIN, Xiaohui SHEN, Jimei YANG, Xin LU, Mohit BANSAL, and Tamara L BERG. "MAttNet: Modular Attention Network for Referring Expression Comprehension". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1307–1315. arXiv: 1801.08186 [cs.CV].

Figure Sources

Ali21 ALIBABA GROUP HOLDING LIMITED. *China Arduino Robot Arm.* `https://www.alibaba.com/countrysearch/CN/arduino-robot-arm.html`. [Online; accesed 16 March 2021]. Mar. 2021.

Asi21 ASIA TIMES ONLINE. *China's robot market is still No. 1.* `https://asiatimes.com/2019/09/chinas-robot-market-still-no-1/`. [Online; accesed 16 March 2021]. Mar. 2021.

Bel+20 Miriam BELLVER, Carles VENTURA, Carina SILBERER, Ioannis KAZAKOS, Jordi TORRES, and Xavier GIRÓ-I-NIETO. "RefVOS: A Closer Look at Referring Expressions for Video Object Segmentation". In: *CoRR* abs/2010.00263 (2020). arXiv: `2010.00263`. URL: `https://arxiv.org/abs/2010.00263`.

Che+17 Liang-Chieh CHEN, George PAPANDREOU, Florian SCHROFF, and Hartwig ADAM. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv* (2017). arXiv: `1706.05587 [cs.CV]`.

Com18 COMPUTER SCIENCE WIKI. *Max-pooling/Pooling — Computer Science Wiki.* [Online; accessed 16 March 2021]. 2018. URL: `%5Curl%7Bhttps://computersciencewiki.org/index.php?title=Max-pooling_/_Pooling&oldid=7839%7D`.

Cor+18 Marcella CORNIA, Lorenzo BARALDI, Hamed R. TAVAKOLI, and Rita CUCCHIARA. "Towards Cycle-Consistent Models for Text and Image Retrieval". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

GQ01 Ramazan GENÇAY and Min QI. "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging". In: *IEEE Transactions on Neural Networks* 12.4 (2001), pp. 726–734. DOI: `10.1109/72.935086`.

HBB19 Radja HACHILIF, Riyadh BAGHDADI, and Fatima BENHAMIDA. "Graduation Thesis Implementing and Optimizing Neural Networks using Tiramisu". PhD thesis. Ecole Nationale Supérieure d'Informatique, June 2019.

IAM21 IAM ROBOTICS, LLC. *The Future of Material Handling.* `https://www.iamrobotics.com/`. [Online; accesed 16 March 2021]. Mar. 2021.

Kaz+14 Sahar KAZEMZADEH, Vicente ORDONEZ, Mark MATTEN, and Tamara L. BERG. "ReferIt Game: Referring to Objects in Photographs of Natural Scenes". In: *EMNLP*. Oct. 2014. URL: `http://tamaraberg.com/referitgame/`.

LeC98 Yann LECUN. *LeNet-5, Convolutional Neural Networks — Personal Website.* `http://yann.lecun.com/exdb/lenet/`. [Online; accessed 15 February of 2021]. 1998.

LKX20 Fei-Fei LI, Ranjay KRISHNA, and Danfei XU. *CS231n: Convolutional Neural Networks for Visual Recognition.* `http://cs231n.stanford.edu/`. [Online; accessed 1 October 2020]. 2020.

Poc19 Pocono Record — Daily Newspaper. *Drone used to track suspects after Rte. 22 crash News*. https://www.poconorecord.com/news/20190311/drone-used-to-track-suspects-after-rte-22-crash. [Online; accessed 12 February of 2021]. 2019.

QDW20 Yanyuan Qiao, Chaorui Deng, and Qi Wu. "Referring Expression Comprehension: A Survey of Methods and Datasets". In: *IEEE Transactions on Multimedia* (2020). arXiv: 2007.09554 [cs.CV].

Rez+19 Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. "Generalized Intersection over Union: A metric and a loss for bounding box regression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2019, pp. 658–666. arXiv: 1902.09630 [cs.CV].

Vas+17 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., June 2017, pp. 6000–6010.

Vey20 Alexander Veysov. "Toward's an ImageNet Moment for Speech-to-Text". In: *The Gradient* (2020).

Wik21a Wikipedia contributors. *Jaccard index — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=1009813550. [Online; accessed 9-April-2021]. 2021.

Wik21b Wikipedia contributors. *Long short-term memory — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1005032489. [Online; accessed 10 March of 2021]. 2021.

Wik21c Wikipedia contributors. *Overfitting — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=1016721642. [Online; accessed 9 April 2021]. 2021.

Yu+18 Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. "MAttNet: Modular Attention Network for Referring Expression Comprehension". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1307–1315. arXiv: 1801.08186 [cs.CV].

Quotation Sources

Ber21 Tim Berners-Lee. *Introduction to Web Accessibility — Accessibility in Context*. https://www.w3.org/WAI/fundamentals/accessibility-intro/. [Online; accessed 6 April of 2021]. 2021.

Cau87 Maureen Caudill. "Neural Networks Primer, Part I". In: *AI Expert* 2.12 (Dec. 1987), pp. 46–52. issn: 0888-3785.

Che+17 Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *arXiv* (2017). arXiv: 1706.05587 [cs.CV].

LeC98 Yann LeCun. *LeNet-5, Convolutional Neural Networks — Personal Website*. http://yann.lecun.com/exdb/lenet/. [Online; accessed 15 February of 2021]. 1998.

Lu+19 Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. “ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks”. In: *arXiv preprint* (2019). eprint: 1908.02265 (cs.CV).

Neu14 John von Neumann. *Common Mistakes in using Statistics: Spotting and Avoiding Them*. https://web.ma.utexas.edu/users/mks/statmistakes/ovefitting.html. [Online; accessed 20 February 2021]. 2014.

Nil09 Nils J. Nilsson. *The Quest for Artificial Intelligence*. 1st. USA: Cambridge University Press, 2009. isbn: 0521122937.

Vas+17 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., June 2017, pp. 6000–6010.

Alphabetical Index

List of topics and terms present in this thesis arranged alphabetically with their corresponding page numbers to locate them within this publication.

This page intentionally left blank.

About the Author

David Álvarez Rosa (Pamplona, 1998) is Mathematics and Industrial Technologies Engineering student at the Polytechnic University of Catalonia (Barcelona, 2016–2021) passionate about maths, artificial intelligence and engineering. He is interested in technology, and is a strong advocate of free (as in *freedom*) software. You can visit him online at:

- Website `https://david.alvarezrosa.com`
- Blog: `https://blog.alvarezrosa.com`
- LinkedIn: `https://linkedin.com/in/david-alvarez-rosa`
- GitLab: `https://gitlab.com/david-alvarez-rosa`
- GitHub: `https://github.com/david-alvarez-rosa`
- Medium: `https://david-alvarez-rosa.medium.com`

In his personal life, he likes doing sport, rides a unicycle, loves the Emacs operating system as a way of life, and is an Esperanto learner. He believes in transparency and openness, in a free world where all citizens must have equal rights.

This page intentionally left blank.

Human-machine interaction is one of the main objectives currently in the field of Artificial Intelligence. This work will contribute to enhance this interaction by exploring the new task of Referring Expression Comprehension (REC), consisting of: given a referring expression—which can be a linguistic phrase or human speech—and an image, detect the object to which the expression refers (i.e., achieve a binary segmentation of the referred object). The multimodal nature of this task will require the use of different deep learning architectures, among them: convolutional neural networks (computer vision); and recurrent neural networks and the Transformer model (natural language processing).

This thesis is presented as a self-contained document that can be understood by a reader with no prior knowledge of machine learning. The bulk of the work consists of an exhaustive study of the REC task: from the applications; until the study, comparison and implementation of models; going through a complete description of the current state of the art. Likewise, a functional, free and public web page is presented in which interaction is allowed in a simple way with the model described in this work.