Master 2

# Reconnaissance et détection des interactions humaines

Stage de fin d'année
M2 SETI
Université Paris-Sud

Présenté et soutenu à Orsay, le 13/09/2018, par :

**OUALI Yassine**

**Encadrement :**

**M. Bertrand Luvison**
Ingénieur chercheur — CEA Saclay
**Mme. Astrid Orcesi**
Ingénieur chercheur — CEA Saclay

**Composition du Jury :**

**M. Alain Mérigot**
Professeur, Responsable du Master — Université Paris-Sud
**Mme. Christine Parey**
Responsable du Master — CEA-INSTN
**M. Abdelhafid Elouardi**
Professeur — Université Paris-Sud
**M. Omar Hammami**
Professeur — ENSTA Paristech

# Detecting and Recognizing Human Interactions

by

**Yassine Ouali**

**Supervisors**

Mr.Bertrand Luvison, Research Engineer - CEA

Ms.Astrid Orcesi, Research Engineer - CEA

**Internship done at CEA LIST, Nano-INNOV - LVIC**
**From April 03 to September 28, 2018**

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Science and Engineering

In the Department of Electronics, Electrical energy and Automatic
School of Engineering, Science and Information Technology
University of Paris-Saclay - University of Paris-Sud


**September 2018**

# Approval

| | |
|---|---|
| **Name:** | **Yassine Ouali** |
| **Degree:** | **M2 Systèmes embarqués et traitement de l'information - SETI (Computer Science and Engineering)** |
| **Title:** | **Detecting and Recognizing Human Interactions** |
| **Examining Committee:** | **Alain Mérigot**<br>University of Paris-Sud<br>Professor<br><br>**Christine Parey**<br>CEA - INSTN<br>Master's Coordinator<br><br>**Abdelhafid Elouardi**<br>University of Paris-Sud<br>Professor<br><br>**Omar Hammami**<br>ENSTA ParisTech<br>Professor |
| **Date Defended:** | **September 13, 2018** |

# Abstract

The ultimate goal of Artificial Intelligence is to create a machine with human-like intelligence, thus giving silicon based computers cognitive abilities similar to what humans have, and maybe one day surpassing them in a form of a utopian machine, an AGI (*Artificial general intelligence*). The long-standing goal of Computer Vision as branch of Artificial Intelligence is imitating the visual system of our brains, going from a row input in the form of photons, transformed in the visual cortex to a rich and deep understanding of the scenes in front of us. To develop such agents, capable of turning pixel intensities into rich semantic information, immense efforts have been put into the field of Computer Vision in the last decades, years ago the field seemed to come to a halt regardless of the amount efforts and research put in, but since 2012 with the resurgence of Deep Learning and simultaneous advances in computing infrastructure and data gathering, significant strides have been made towards this goal over the last few years, now computers are capable of classifying images into different subcategories (*e.g. types of boats*), labeling each pixel of an image and predicting the human pose with real time performances. However, most of this progress consists of assigning few labels to a set of elements of an image, and to obtain a visual understanding of the world around us through the lenses of a camera, we must leap forward and take it a step further.

In this work we address the problem of detecting and recognizing Human-Interactions in still images, considering that the significant actions observed in a scene are often a result of human actions and interactions. We tackle the problem of detecting human interactions in the form of triplets *<subject, verb, object>*. In particular, first we propose a new dataset for Human-Human Interactions containing 5500 images, 25 interactions and 80 object categories, this dataset is by far, to our knowledge, the biggest one available for this particular problem. Second, based on our hypothesis that the type of objects in an image, their relative positions, their appearances and features give strong cues for detecting the interactions taking place. We develop a one-stage model capable of doing joint object-detection and interaction-recognition. We argue that with our one-stage approach, we can obtain higher inference times, constant computation complexity and better real time performances, we validate our approach on the new proposed dataset where we show compelling results. Finally, we propose to address the problem of detecting all types of interactions (*i.e. Human-Human* and *Human-Object* interactions) as the new path forward for future work and we provide a new baseline for it.

**Keywords:** Computer Vision, Machine Learning, Deep Learning, Human interactions, Object Detection, Data Gathering, Artificial Inteligence.

# Acknowledgements

There are many people I must thank for contributing to the six wonderful months of my experience as an intern in CEA LIST.

First I must thank my advisors, Mr.Bertrand Luvison and Ms.Astrid Orcesi who, through many meetings, dicussions and emails over six months, helped me to understand many things, acquire various new skills, and most of all, teaching me many aspects of computer vision and machine learning. Their passion, kind-heartedness, foresight, ambition and zeal for perfection are infectious. Always giving me great advices and better ideas. Whenever I produced terrible results they patiently and kindly pointed out the erroneous parts. Always capable of uncovering the more fundamental story and placing it in a wider context. I will always be very grateful to them.

My thanks also go to my colleagues. In particular, the trainees, doctoral students, engineers, researchers, interns, and secretaries of the LVIC laboratory, for their professionalism, their conviviality and their communicative good humor and for having thus formed an efficient and pleasant working environment. I would particularly like to thank the members of the Scene Analysis team with whom I have had many enriching exchanges and experiences.

I would especially like to thank Ms.Camille Dupont for the help she provided me with on the web developpement side of this work. Ms.Sanaa Chafik for answering the questions of a confused intern, and offering a continus stream of advice each time I get stuck on a technical problem. And Mr.François Gravel and Mr.Giani Franchi, with whom I shared the office space, for all the interesting discussions we had, ranging from TensorFlow to the latest Marvel movies, and the numerous technical help they provided me with.

My thanks goes to Mr.Quoc Cuong PHAM, scene analysis team leader, for providing such a pleasant working environment with the quality of his leadership, and for the numerous remarks and helpful suggestions he provided me with during the presentations and the review of this work. And to Mr.Patrick Sayd, head of the LVIC laboratory, who warmly welcomed me his unit.

And I would also like to express my sincere gratitude to all the administrative and teaching staff of M2 SETI at University Paris-Sud, for their considerable efforts and their outstanding teaching capabilities, especially Mr.Alain Mérigot and Ms.Christine Parey for their remarkable work throughout the Master's year. My warm thanks also go to the members of the jury for their interest in my project, by agreeing to review my work and enrich it with their proposals. And to all people who I had the pleasure of interacting with and learning from during my Master's degree.

*To all the people who may have contributed to this work, directly or indirectly, I am grateful to you.*

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Context

> *"We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried."*
>
> — Alan Turing, *Computing Machinery and Intelligence (1950)*

One of the main objectives of Artificial Intelligence, as mentioned in the inspiring quote by the father of modern computing Alan Turing, is giving computer the ability to "see", understand the visual world and enabling them to communicate it affectively with us.

Humans have an intrisic and an intuitive capabilitiy to accomplish a wide variety of tasks quite easily, tasks that involve complex visual recognition and scene understanding, tasks that involve communication in natural language and tasks that combine information of various modalities to extract a meaning and semantic understanding. For instance, a quick glance at an image is suffcient for a human to point out and describe an immense amount of details about the visual scene. Using the examples in Figure 1.1, we can look at the two images and immediately point out and describe that: "We see two persons in the image", "Both persons are young, one is more so than the other" or simply "Two persons are hugging each other" for the first image, and "A brown dog is spotted", "A blue skateboard with blue wheels" or simply describe the entire image as a "dog riding a skateboard".

**Challenges:** Since this ability feels so natural and effortless for us it can be easy to forget how difficult this task is for a computer. In a computer, this image is represented as one large array of numbers indicating the brightness at any position. An ordinary image might have a few million of these pixels and a computer must transform these patterns of brightness values into highlevel, semantic concepts such as a "dog". Moreover, a different breed of dogs seen under different lighting conditions, with a different camera angle, or in a different pose might still depict a "dog riding a skateboard", but the pattern of brightness values could be completely different. Conversely, patterns with very similar low-level statistics

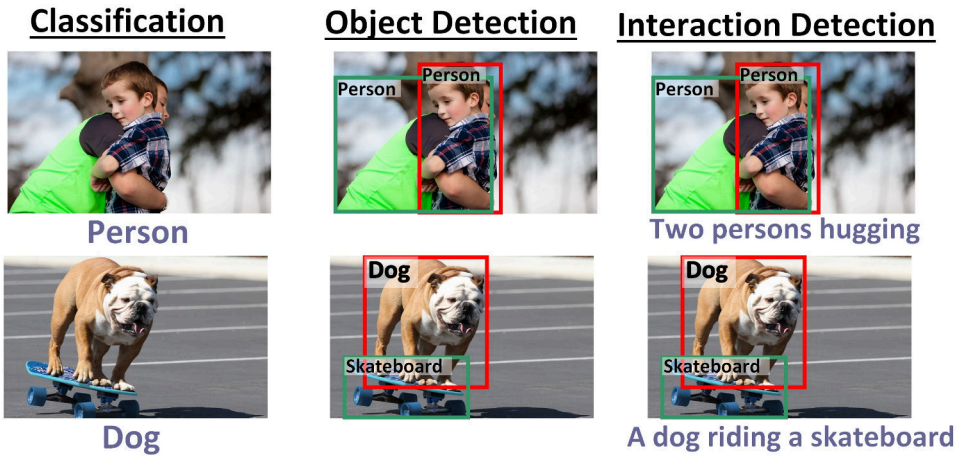



Figure 1.1: Visual recognition tasks. **Left**: Assigning a category to an image. **Middle**: Detecting the objects in an image. **Right**: Describing the interactions taking place in an image with a sentence.

(e.g. fur-like high frequency patterns) might instead be part of many different objects (carpets, coats, etc.) or animals (cats, bears, monkeys, etc.). Therefore, the very natural task of pointing out and naming different parts of an image and the interactions taking place in fact involves a complex pattern recognition process of identifying salient subsets of a grid of a few million brightness values and annotating them with sequences of integers. Moreover, detecting the interactions taking place often require detecting and describing complex high-level concepts that are not only visual but require difficult inferences. For example, some images can be annotated by humans as "a group of men fighting", which requires the ability to detect multiple people and analyze their poses, spatial arrangements or even their facial features. Alternatively, someone could be described as "waiting" for something, "playing" with something, or "talking" to someone.

Despite the difficulty of this task, we have recently witnessed rapid progress in the area of visual recognition. In particular, the state of the art image recognition models based on deep convolutional neural networks [60] have become capable of distinguishing thousands of visual categories at accuracies comparable to humans, or even surpassing them in some fine-grained categories such as breeds of dogs [42]. The progress on related tasks such as segmentation and object detection has been similarly dramatic [49, 62]. Together, these advances have enabled many real-world applications including face detection and recognition, personal photo search, perception in robotics and self-driving cars, etc.

However, the predominant approach in most of these applications is to model the visual recognition problem as a task of classifying images into some number of fixed and hard-coded visual categories (e.g. Figure 1.1, left). For instance, the ImageNet visual recognition challenge [42] (a popular visual recognition benchmark) consists of a set of 1,000 categories that were picked manually by the organizers (examples include "hot dog", "screwdriver", "jellyfish", "notebook" and also exclude many common concepts such as "person", "face", etc.). Similarly, for object detection where the objective is to not only classify the main object in the image but all the objects with a bounding box around each one (e.g. Figure 1.1, middle), PASCAL VOC [12] is one of the benchmarks for object detection, it uses a different set of 20 manually chosen categories (e.g.. "car", "person", "potted plant", etc.). While visual categories constitute a conve-

nient modeling assumption, the task of image classification and object detection is easier in comparison to the complexity of descriptions that humans can compose for images such as detecting the interactions taking place (e.g. Figure 1.1, right).

**Our focus:** Current state of the art on action recognition consists of classifying a video clip containing the action, or marking a bounding box around the approximate location of the agent doing the action. Most current action recognition datasets classify each person into doing one of $k$ different activities and focus on coarse activities (e.g. "playing baseball", "cooking", "gardening"). Such a coarse understanding is incomplete and a complete visual understanding of an activity can only come when we can reason about fine grained actions constituting each such activity (e.g. the person on the left is "hitting" the one on the left, two persons are "hugging", they are "punching" each other, etc.), reason about people doing multiple such actions at the same time, and are able to associate objects in the scene to the different semantic roles for each of these actions. To overcome some of these limitations, our main focus in this work is Human Interactions (i.e. Human-Human direct interactions), we first introduce a dataset containing 5500 images with 25 annotated interactions and 80 annotated objects, this dataset is by far the largest one for this type of interactions by a large margin, with the help of this dataset, we aim to develop a model capable of doing joint object detection and interaction detection, the output will be a set of triplets for all the interactions taking place in the image (<*subject, verb, object*>) together with bounding boxes over the subjects/objects in question. In other words, our goal is to extract highlevel semantic concepts in the form of triplets describing the main actions of an image and enabling computers to translate static image into rich decriptions, taking the next step toward connecting the two modalities of vision and natural language.

**Motivations.** For the **short-term** motivations our goal can be motivated with more concrete, short-term and practical arguments. First, detecting Human-Interactions inherits of many other visual recognition tasks that are currently treated as separate areas in computer vision, including object recognition and detection, scene classification, attribute classification, action recognition, etc. Second, many applications can benefit from detecting Human-Interactions, security applications such as crowd surveillance, image description based search, or even for a special case of image Captioning and Visual Question Answering. For the **long-term**, these techniques are a step towards a future in which we can interact with computers in natural language, In addition, working towards artificially intelligent agents will require us to make available to computers large amounts of information about how our world works: the physical domain that contains information about the world, scenes, objects and interactions and the digital domain of the Internet that contains a vast amount of semantic information that cannot be inferred from physical domain alone (e.g. what happened in 1760), encoded primarily in natural language. Therefore, vision and language are the primary channels by which the knowledge of the world can be accessed and it is critical that we develop techniques that can relate information across the two domains instead of processing each independently.

## 1.2 Presentation of the company

The internship took place in CEA LIST research center for technological progress, at the Nano-INNOV location and more specifically within LVIC (Vision and Content Engineerings) lab as part of the Scene

Analysis team, for approximately 6 months, from **3rd April** to **28th September, 2018**.

CEA, as public an industrial and a commercial establishment, is a major player in research, development, and innovation. It intervenes in three major areas:

- Global Defense and Security
- Carbon-free energies
- Technologies for information and health

CEA's scientific skills range from basic research to technological research, the latter relying on the exploitation of exceptional equipment such as supercomputers, research reactors, power lasers, etc. All these elements encourage a real involvement of the CEA in the national and international industrial and economic fabric. It has also a homogenous presence in France with 16,000 employees and a budget of 4 billion euros. With ten centers to develop and maintain numerous partnerships with other research organizations, universities, and local communities in a strong communication environment to foster knowledge transfer.

In addition, all these centers now have more than 1600 active priority patents and have enabled the creation of 120 new companies since 1984. The French government is now advised by the CEA for questions of external nuclear policy. Allowing CEA to become increasingly involved in the European and international space by representing France to international organizations in the nuclear sector, by facilitating and developing cooperation with counterpart organizations in other countries as a whole.

### 1.2.1 Presentation of the host laboratory

Located in South of Île-de-France. CEA LIST is the **Laboratory for Integration of Systems and Technologies** of CEA's Technological Research Department. CEA LIST's research focuses on software-intensive systems. Its activities revolve around three themes presenting strong societal and economic challenges: Embedded Systems, Interactive Systems and Sensors and signal processing.

CEA LIST **Vision and Content Engineering Laboratory** employs 40 researchers and engineers working on the analysis and interpretation of image, text, video and multimedia data. They conduct research on the analysis and interpretation of multimedia and multilingual data, and other computer vision related applications such as augmented reality, industrial vision control, video surveillance.

The scientific issues are two folds:

- Develop efficient and robust algorithms for content analysis by extraction, classification, semantic analysis of the media. The reconstruction or fusion between these data then allows the interpretation of scenes or documents.
- Develop the methods and tools for building, formalizing and organizing the knowledge needed for these algorithms.

These issues are addressed through the four scientific themes of the laboratory:

- 3D Perception and Mobility: 3D vision reconstruction for dimensional control and 3D localization applications for augmented reality applications and geolocation systems.

- Video analysis: Video analysis for video protection applications and, more generally, for video assistance systems (driver assistance systems: detection of people, monitoring of vulnerable people,. . . ).
- Text analysis: multilingual semantic analysis of documents for indexing, search, standby and filtering applications.
- Fusion Multimedia: indexing and searching multimedia information for searching large databases or mobile applications.

In addition, the laboratory has established several joint collaborations with manufacturers. Such as collaborations with Thales for video surveillance, ARCure for monitoring the safety pedestrians, and Valeo for 3D vehicle analysis.

### 1.2.2 Internship progress

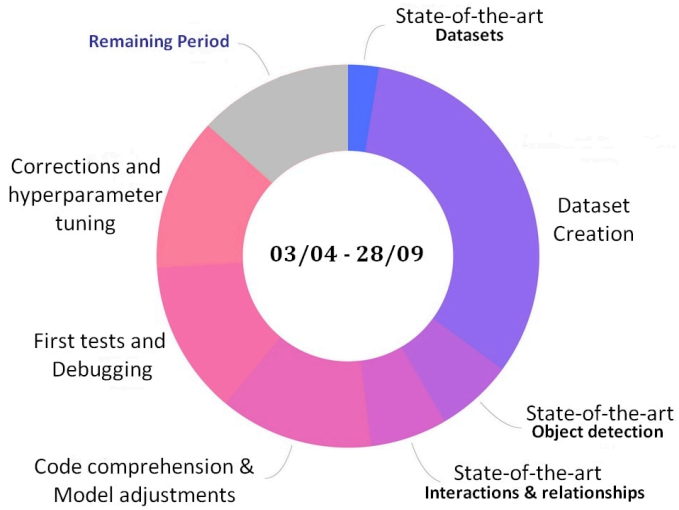



Figure 1.2: Showing roughly how the progress of the internships went, neither the partitioning nor the subjects are exact but just a mere estimate.

As shown in Figure 1.2, the internship began with a quick overview of the existing datasets for Human-Interactions to get an idea of their quality, after a thorough assessment, it become clear that a dataset for these types of applications was needed, with better quality images, greater diversity in terms of the interactions and more precise annotations. By knowing the type of dataset we want and its characteristics, the dataset creation phase began, after combining the existing datasets, adding annotated images from other types of datasets and collecting new images form the web, we designed a new annotation tool for adding the interactions as triplets, and finally the interaction triplets were added for a total of 5500 images. The next phase was a deeper look into the state-of-the-art object detection, relationships and interactions recognition systems to draw inspiration for desinging a new architechture, followed by a first implementation of the model and minor adjusments for detecting Human-Interactions, ending up with a series of corrections, debugging and hyperparameter tuning to obtain better performances.

## 1.3 Contributions and Outline

In this work we introduce a new dataset for Human-Interactions and we develop a model for detecting human interactions in still images. In particular, we develop a neural network architecture capable of detecting all the objects present in the image (from 80 categories) and more importantly the interactions taking place in the form of triplets (<*subject, verb, object*>).

In **Chapter 2** we provide a general overview of the field of Computer Vision and Machine Learning with the relevant background for Deep Learning, more specifically for supervised learning, backpropagation, optimization, neural networks (i.e. convolutional neural networks), and describe commonly used architectural design patterns for image classification.

In **Chapter 3** we provide a detailed state-of-the-art for object detection, visual relationships detection and human interactions detection and recognition with an overview of the datasets used in these types of applications.

In **Chapter 4** we go through the process of creating the Human-Interactions dataset, by combining existing Human-Interactions datasets, adding images from other types of datasets and collecting new images from the web. Annotating all the objects (from 80 pre-defined categories) in the images, creating a custom annotation tool for describing the interaction in the form of triplets (<*subject, verb, object*>) and annotating the interactions.

In **Chapter 5** we develop a model for joint object detection and human interaction recognition. That is, given a static image the model is capable of outputing a set of triplets for detecting human interactions, each subject/object is detected in the image using a bounding box around it, The model will first process the image to detect all the objects in it, and then use these detections to infer the interactions taking place between them. We then present a possible problem extension, going from detecting either Human-Human or Human-object interactions to detecting both types of the interactions, and we propose a first benchmark to be used for detecting both types of interactions.

Finally, in **Chapter 6** we identify the remaining challenges and discuss the path forward.

# Chapter 2

# Overview

This chapter provides an overview of the field of computer vision and the necessary technical background on machine learning, neural networks and the kind of software used, a large portion of this chapter was inspired by Stanford's CS231n Course, Deep Learning book from Goodfellow et al. [20] and Hands-On Machine Learning with Scikit-Learn and TensorFlow from Aurélien Géron [22].

## 2.1 Computer Vision

Computer Vision has a dual goal [23]. From the biological point of view, computer vision aims to come up with computational models of the human visual system. From the engineering point of view, computer vision aims to build autonomous systems which could perform some of the tasks which the human visual system can perform (and even surpass it in many cases), the two goals are intimately related. The properties and characteristics of the human visual system often give inspiration to engineers who are designing computer vision systems. Conversely, computer vision algorithms can offer insights into how the human visual system works.

When computer vision first started out in the early 1970s [51], it was viewed as the visual perception component of an ambitious agenda to mimic human intelligence and to endow robots with intelligent behavior. Early attempts at scene understanding involved extracting edges and then inferring the 3D structure of an object from the structure of the 2D lines. In the 1980s, a lot of attention was focused on more sophisticated mathematical techniques for performing quantitative image and scene analysis, troughtout the 1990s ans 2000s the field has continued to see a deepening interplay between the vision and graphics fields.

In 2006, Geoffrey Hinton et al. published a paper [16] showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%). They branded this technique "Deep Learning." Training a deep neural net was widely considered impossible at the time, and most researchers had abandoned the idea since the 1990s. This paper revived the interest of the scientific community and before long many new papers demonstrated that Deep Learning was not only possible, but capable of great achievements that no other Machine Learning (ML) technique could hope to match, thanks to the increase in computational power, the amount of available training data and the introduction of new methods for training deep nets. One of the fields that benefited the most from the resurgence of Deep Learning is Computer Vision, after the introduction of AlexNet [3] and the 10% error gain it showed on the 2012 ILSVRC (The ImageNet Large Scale Visual Recognition) Challenge [42], Deep

Learning techniques now dominates a lot of the computer vision research and it is applied to several visual recognition tasks (Figure 2.1[1]);

- Image Classification: Classify an image based on the dominant object inside it,
- Object Localisation: Predict the image region that contains the dominant object and classify it,
- Object Detection: Localize and classify all objects appearing in the image. includes: proposing regions then classify the object inside them,
- Semantic Segmentation: Label each pixel of an image by the object class that it belongs to, such as person, dog and handbag in the example,
- Instance Segmentation: Label each pixel of an image by the object class and object instance that it belongs to,
- Keypoint Detection: Detect locations of a set of predefined keypoints of an object, such as keypoints in a human body, or a human face.

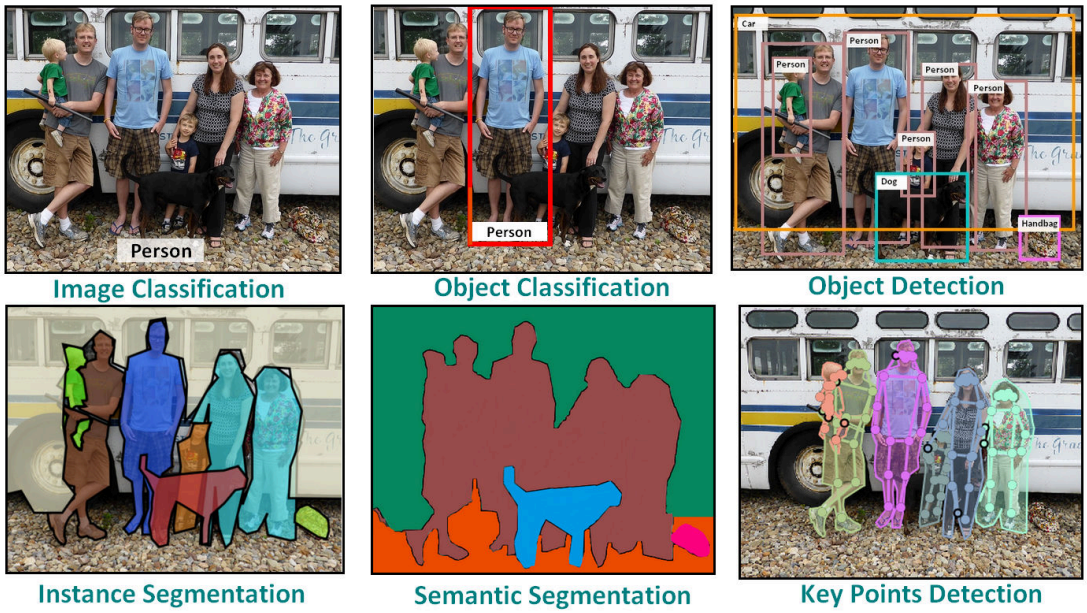



Figure 2.1: Various visual recognition tasks that benefited the most from the application of Deep Learning techniques .

## 2.2 Supervised Learning

Many practical problems can be formulated as requiring a computer to perform a mapping $f : X \to Y$, where $X$ is an input space and $Y$ is an output space [32]. For instance, in visual recognition $X$ could be the space of images and $Y$ could be the interval $[0, 1]$ indicating the probability of a cat appearing somewhere in the image. but in many cases it is difficult to manually specify the function $f$ by conventional means. The supervised learning paradigm offers an alternative approach that takes advantage of the fact that it is often relatively easy to obtain examples $(x, y) \in X \times Y$ of the desired mapping.

[1] A picture from COCO dataset [56] : **Link**

**The objective**. For a training dataset of $n$ examples $\{(x_1, y_1), ...(x_n, y_n)\}$ made up of independent and identically distributed (i.i.d.) samples from a data generating distribution $D$; i.e. $(x_i, y_i) \sim D$ for all $i$. We then think about *learning* the mapping $f : X \to Y$ by searching over a set of candidate functions and finding the one that is most consistent with the training examples. More precisely, we consider some particular class of functions $\mathcal{F}$ and choose a scalar-valued loss function $L(\hat{y}, y)$ that measures the disagreement between a predicted label $\hat{y}_i = f(x_i)$ for some $f \in \mathcal{F}$ and a true label $y_i$. Our objective in learning is to find $f^* \in \mathcal{F}$ that ideally satisfies:

$$f^* = \arg\min_{f \in \mathcal{F}} E_{(x,y)\sim D} L(f(x), y) \tag{2.1}$$

Unfortunately, the optimization problem above is intractable because we do not have access to all possible elements of $D$ and therefore cannot evaluate the expectation. However, under the i.i.d. assumption we can approximate the expected loss in Equation 2.1 above with sampling by averaging the loss over the available training data:

$$f^* \approx \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) \tag{2.2}$$

**Regularization**. Optimizing Equation 2.2 instead of Equation 2.1 poses challenges. For instance, a function $f$ that maps each $x_i$ in the training data to its $y_i$ but returns zero everywhere else. This would be a solution to Equation 2.2 but we would expect very high loss for all other points in D that are not in the training set. In other words, we would not expect this function to generalize to all $(x, y) \sim D$. An additional concern is that there may be many different functions that all achieve the same loss under Equation 2.2, but their generalization outside of the training data could vary. Both of these concerns can be alleviated by introducing a regularization term R to the objective:

$$f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) + R(f) \tag{2.3}$$

where $R$ is a scalar-valued function that encodes preference for some functions over others, regardless of their fit to the training data. This addition can be partly justified as following the principle of Occam's razor, which could be stated as: *"Suppose there exist two explanations for an occurrence. In this case the simpler one is usually better",* and its intended effect compensate to some extent for the discrepancy between the objective in Equation 2.1 and Equation 2.2.

**Summary**. In supervised learning we are given a dataset of $n$ datapoints $\{(x_1, y_1), ...(x_n, y_n)\}$ where $(x_i, y_i) \in X \times Y$ and we identify three quantities to formalize the problem (see Figure 2.2):

1. The search space of functions $\mathcal{F}$, where each $f \in \mathcal{F}$ maps $X$ to $Y$.
2. The scalar-valued loss function $L(\hat{y}, y)$ that evaluates the mismatch between a true label $y$ and a predicted label $\hat{y} = f(x)$.
3. The scalar-valued regularization loss $R(f)$ that measures the complexity of a mapping.

In deep learning the space of functions $\mathcal{F}$ will be a neural network with some parameters (weights $W$ and biases $b$), the loss $L$ will be a euclidean loss in regression (i.e. $(\hat{y} - y)^2$) or a cross-entropy loss in classification (i.e. $\sum_{k=1}^{K} y_k log(\hat{y}_k)$), etc. And the regularization $R$ is most commonly the L2 norm (i.e. sum of squares of all weights).

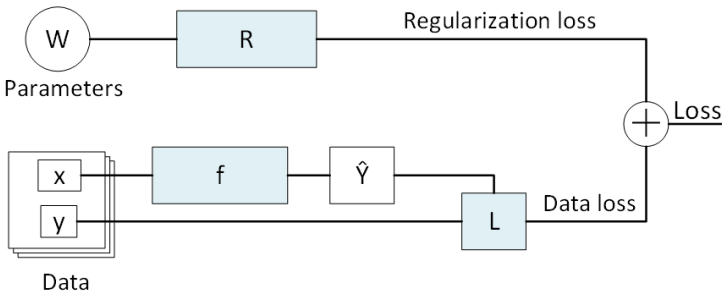



Figure 2.2: Diagram of the data flow in a typical supervised learning problem approached with a neural network. The input is a dataset of pairs $(x, y)$ of examples $x$ and labels $y$. With three functions: 1) The function $f$ that maps the examples $x$ to some predicted labels $\hat{y}$ (a neural network in our case). 2) The function $L(\hat{y}, y)$ that evaluates the mismatch between the prediction and the true label, and 3) the regularization function $R$ that evaluates the complexity of the mapping. The objective becomes to find the parameters $\theta$ that minimize the final loss.

## 2.3 Optimization

In the last section we saw that we can reduce the task of learning a model for a supervised learning problem to solving an optimization problem of the form $\theta^* = \arg\min_\theta g(\theta)$, where $\theta$ is a parameter vector and $g$ usually combines the average loss of all examples and a regularization penalty.

**Derivative free optimization.** Based on the fact that we can evaluate $g(\theta)$ for any arbitrary $\theta$ so one approach is to draw a large number of $\theta$ at random from some distribution, check each one, and take the one that minimizes $g$. A more elaborate approach might iteratively seek to improve some candidate $\theta$ by repeatedly making small perturbations. Unfortunately, typical neural networks we want to train might have parameter vectors with several million or billion parameters, so many of these approaches are computationally intractable.

**First order methods.** We can improve the effciency of the optimization by making additional assumptions about $g$. By using differentiable functions then we can compute the gradient $\nabla_\theta g$ with backpropagation. The gradient is a vector of partial derivatives, giving us the slope of $g$ along every dimension of $\theta$. The gradient allows us to construct the first order approximation in the Taylor expansion of $g$. We can use the gradient as a search direction; in particular, we can improve $\theta$ (to achieve a lower $g$) by adding to it a small amount of the negative gradient direction. This insight motivates the gradient descent (GD) algorithm that alternates the two steps: 1) evaluate the gradient with backpropagation and 2) update the parameters by taking a small step in the direction of the negative gradient. As a last practical consideration, the datasets used in practice can be very large (e.g. ImageNet has 1 million training images) so we only estimate the gradient using a small minibatch of examples (e.g. around 16/32/48 etc.) at a time. This allows us to perform many approximate updates instead of fewer exact updates - a strategy that works well in most practical applications. A critical parameter in Stochastic Gradient Descent (SGD) is the step size $\epsilon$ (also called the *learning rate*). If it is too high the optimization may not converge or even diverge. The resulting algorithm, SGD is summarized in Algorithm 1:

**Advanced first order optimization techniques.** In practice one can often obtain faster converge by modifying the computation of the update direction (Step 3 in Algorithm 1). One of these techniques is the **Momentum** update, designed to encourage progress along small but consistent directions of the gradient. Using a shorthand notation of $\boldsymbol{g} = \nabla_\theta g(\theta)$ for the gradient vector, the update $\nabla\theta$ is computed

**Algorithm 1** Stochastic Gradient descent.

1: **Given** a starting point $\theta$
2: **Given** a step size $\epsilon \in R^+$
3: **repeat**
4: 1. Sample a minibatch of $m$ examples $\{(x_1, y_1), ..., (x_m, y_m)\}$ from training data
5: 2. Estimate the gradient $\nabla_\theta g(\theta) \approx \nabla_\theta \frac{1}{m}[\sum_{i=1}^{m} L(f_\theta(x_i), y_i) + R(f_\theta)]$ with backpropagation
6: 3. Compute the update direction: $\triangle\theta = -\epsilon\nabla_\theta g(\theta)$
7: 4. Perform a parameter update: $\theta := \theta + \triangle\theta$
8: **until** convergence.

first by updating an intermediate variable $\boldsymbol{v} := \alpha\boldsymbol{v} + \boldsymbol{g}$ (initialized at zero), and then computing the update as $\nabla\theta := -\epsilon\boldsymbol{v}$. notice that the variable v contains an exponentially-decaying sum of previous gradient directions. This update has a close relationship to physics, where the gradient is interpreted as a force $F$ on a particle with position $\theta$, incrementing its velocity instead of its position directly. This is consistent with Newton's second law $\boldsymbol{F} = \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t}$, where momentum $\boldsymbol{p} = mv$, and the mass $m$ is assumed to be a constant.

A number of methods have also been developed that modulate the update using the second moment like **Adagrad** [26] and **RMSProp** [53] and **Adam** [33] that estimates both first and second running moments.

**Cross-validation.** The above problem formulation and the optimization require many settings like the regularization strength $\lambda$, the step size for stochastic gradient descent $\epsilon$, the number of hidden units in a neural network, etc. Many of these parameters are difficult to attach to the parameter vector $\theta$ and train with gradient descent. Instead, we resort to stochastic optimization techniques; by trying several possibilities, optimize the model in each case and finally evaluate the predictions on a withheld, validation set of examples that were not used during training. This is a way of estimating the generalization error.

## 2.4 Backpropagation

In the process of finding the mappings $f \in \mathcal{F}$ that achieve low regularization cost and map $X$ to $Y$ consistent with the training data, we need to evaluate the gradient of the loss function and then use stochastic gradient descent to minimize it.

Backpropagation is the process by which we effciently compute gradients of scalar valued functions with respect to their inputs. The backpropagation algorithm is a recursive application of the chain rule from calculus. The function we are interested in computing gradients of is $g$, which takes as input the dataset of examples $(x_i, y_i)$ and the parameters $\theta$. We are specifically interested in the gradient $\nabla_\theta g$ with respect to the parameters $\theta$ in order to perform the parameter update.

**General notation.** With an input vector $x_0$ that we transform through a series of functions $x_i = f_i(x_{i-1})$ where $i = 1, ..., k$ and the last $x_k$ is a scalar. Assuming that the gradient exists, we calculate the Jacobian matrix $\frac{\partial x_i}{\partial x_{i-1}}$ of all intermediate transformations, which tells us how every output dimension of $x_i$ depends on every input dimension of $x_{i-1}$. By chain rule the final gradient we are interested in is simply the matrix product of all the Jacobians: $\frac{\partial x_0}{\partial x_k} = \prod_{i=1}^{k} \frac{\partial x_i}{\partial x_{i-1}}$.

It's important to note that in most neural network applications $x_0$ is large (e.g. all pixels of an image) while $x_k$ is small (e.g. number of classes). Therefore, it becomes computationally important to evaluate the above product of all Jacobians from end to front - i.e. starting with $\frac{\partial x_k}{\partial x_{k-1}}$ down to $\frac{\partial x_1}{\partial x_0}$. Conversely, if there was a single input and many outputs then it would be more efficient to go front to back (*forward-*

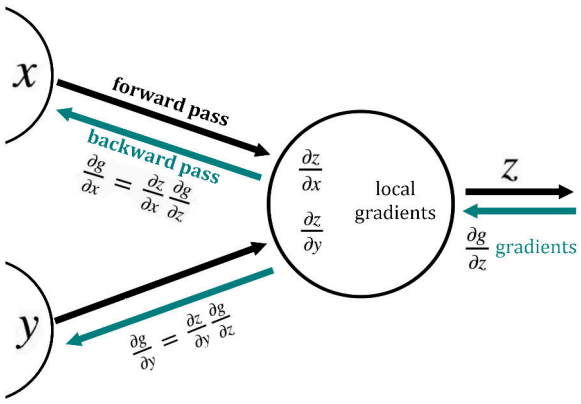



Figure 2.3: An example of backpropagation along a computational graph. During forward pass $x$ and $y$ take on specific (numerical) values and the vector (or scalar) $z$ is computed using some fixed function (e.g. $z = x \odot y$). The value $z$ goes off into a computational graph and eventually at the end of the graph the total loss $g$ (a scalar) is computed. The backward pass proceeds in the reverse order, recursively applying the chain rule to find the influence of all inputs of the graph on the final output. In particular, this computational unit finds out what $\frac{\partial g}{\partial z}$ is, telling us how $z$ influences the final graph output. The chain rule states that to backpropagate this we should take the global gradient on $z$, $\frac{\partial g}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial g}{\partial z}$ and multiply it onto the local gradients for each input.

*mode differentiation*). Unfortunately, reverse mode differentiation requires us to keep all intermediate values computed during the forward pass in memory because we need them as we backpropagate the gradients backward later - this is a price to pay in memory capacity for computational effciency (in most cases Jacobian matrix is almost entirely zero except for elements along the diagonal, thus we can only store the diagonal elements and the matrix multiplication can then be implemented very effciently by only doing an elementwise multiplication with the elements along the diagonal).

In summary, evaluating the gradient of the output with respect to the input reduces to, by the chain rule, a product of Jacobian matrices. In neural network applications we first perform a *forward pass* in which we take a batch of data $\{(x_i, y_i)\}_{i=1}^m$ and current parameters $\theta$ and "forward" the network to compute all intermediate values (caching them for later) and the cost $g$. Then during backpropagation we proceed backwards through all intermediate stages (the "backwards pass") and "chain" the local gradients, each time matrix multiplying by the next Jacobian matrix in the full product. In many cases we do not need to actually create the full Jacobian matrices and can take advantage of special structure to chain the gradients in a more computationally efficient manner.

## 2.5 Neural networks

In the previous sections we saw that we can define arbitrary differentiable functions $f$ that transform the inputs $x$ to predicted outputs $\hat{y}$, and optimize the model with respect to any differentiable loss function using stochastic gradient descent. We now turn to the details of the function $f$.

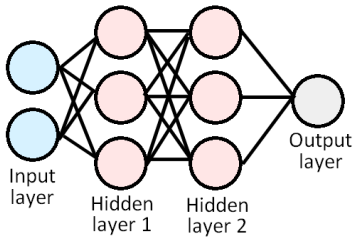



Figure 2.4: An example arrangement of neurons in a 3-layer neural network. Neurons in one layer have connections to all neurons in the previous layer but are not connected to each other. This arrangement allows us to efficiently evaluate activations of all neurons in one layer with a matrix multiplication.

### 2.5.1 Feed Forward Neural Networks

In absence of any assumptions about the structure of $x$, we construct neural networks by repeating matrix multiplications and element-wise non-linearities. As an example, a 2-layer neural network would be implemented as $f(x) = W_2\sigma(W_1x)$, where $W_1$, $W_2$ are matrices and $\sigma$ is an element-wise non-linearity (e.g. tanh). A 3-layer (Figure 2.4) network would have the form $f(x) = W_3\sigma(W_2(\sigma(W_1x)))$, etc, but if the non-linearity is an identity function then the entire neural network reduces to a linear function (i.e. a 1-layer neural network is a simple linear transformation). Common settings for the non-linearities are tanh, the sigmoid function $\frac{1}{(1+e^{-x})}$ and the rectified linear unit (ReLU) $\max(0, x)$. Normally the last layer of the neural network does not contain any non-linearity.

### 2.5.2 Convolutional Neural Networks

Convolutional[2] Neural Networks (CNNs, or ConvNets) [60] are neural network architectures specifically designed for handling data with some spatial topology (e.g. images, videos, sound spectrograms, or character sequences in text). In each of these cases an input example $x$ is a multi-dimensional array (i.e. a tensor). E.g. a $256 \times 256$ color image is a $256 \times 256 \times 3$ tensor (for 3 color channels red, green, blue). A sound spectrogram could be an array of size $1000 \times 128$ indicating the amplitude of any one of 128 frequencies at any point in time from $t = 1, ..., 1000$. In many of these cases the input dimensionality is high (e.g. $256 \times 256$ color image will have approximately 200,000 numbers) and it is wasteful in both number of parameters and processing time to use fully connected layers. In these cases we prefer to design neural network architectures that are aware of the spatial layout of the input and use specific local connectivity and sensible parameter sharing schemes. In this work we are primarily concerned with processing images.

A convolutional layer for images (i.e. assuming input tensors with three spatial dimensions):

- Accepts a tensor of size $W_1 \times H_1 \times D_1$
- Requires 4 hyperparameters: The number of filters $K$, their spatial extent $F$, the stride with which they are applied $S$, and the amount of zero padding on the borders of the input $P$.
- The convolutional layer produces an output volume of size $W_2 \times H_2 \times D_2$, where $W_2 = (W_1 - F + 2P)/S + 1$, $H_2 = (H_1 - F + 2P)/S + 1$, and $D_2 = K$.

[2]The operation used in a convolutional neural network does not correspond precisely to the definition of convolution in other fields such as Mathematics.

- The number of parameters in each filter is $F * F * D1$, for a total of $(F * F * D1) * K$ weights and $K$ biases, the spatial extent of the filters is small in space $(F * F)$, but always goes through the full depth of the input tensor $(D1)$.
- In the output tensor, each $d - th$ slice of the output (of size $W_2 \times H_2$) is the result of performing a valid convolution of the $d - th$ filter over the input tensor with a stride of $S$ and then offsetting the result by $d - th$ bias.

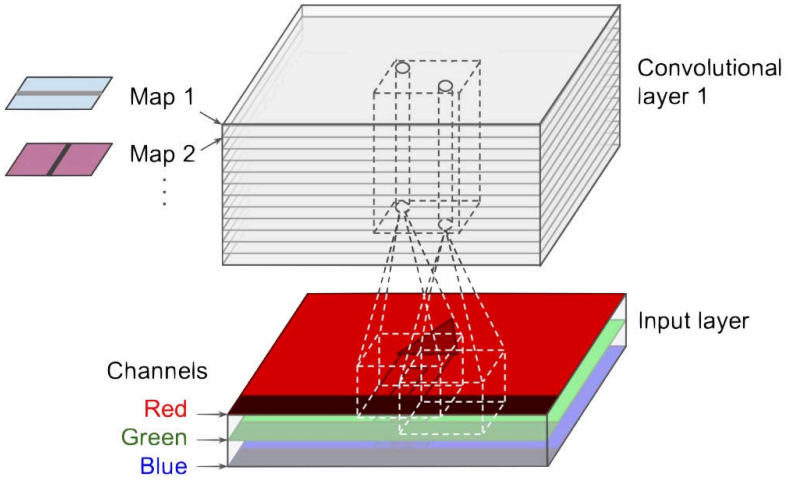



Figure 2.5: Illustration (taken from [22]) of convolving a set of filters (which we will eventually learn) over an input image, The filters are always small spatially compared to the input, but always span the full depth of the input array (3). Each convolution produces an activation map, where each element is the result of a dot product between the filter and the input. A convolutional layer has not just one but a set of different filters, each applied in the same way and independently, resulting in their own activation maps. The activation maps are finally stacked together along depth to produce the output of the layer, where each one detects different features.

In general, the more convolution steps there is, the more complicated features the network will be able to learn to recognize. E.g. in Image Classification a ConvNet may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to deter higher-level features, such as facial shapes in higher layers [66].

**Pooling layers.** In addition to convolutional layers, to further control overfitting it is common to use pooling layers that decrease the size of the representation with a fixed downsampling transformation and add location invariance, the pooling layers operate on activation map independently and downsample them spatially. A commonly used setting is to use $2 \times 2$ filters with stride of 2, where each filter computes the max operation (i.e. over 4 numbers). The result is that an input tensor is downscaled exactly by a factor of 2 in both width and height and the representation size is reduced by a factor of 4 at the cost of losing some local spatial information.

**ConvNet architectures.** A convolutional network is built by stacking convolutional layers and possibly introducing pooling layers to control the computational complexity of the architecture. A typical convolutional neural network architecture that processes images might take the form [INPUT, [CONV, CONV, POOL] $\times 3$, FC, FC]. Here, INPUT represents a tensor of a batch of images (e.g. $[100 \times 32 \times 32 \times 3]$ for a batch of $100, 32 \times 32$ color images) CONV is a convolutional layer with $3 \times 3$ filters applied with padding of 1 and stride of 1, POOL stands for a typical $2 \times 2$ filter max pooling layer with stride of 2,

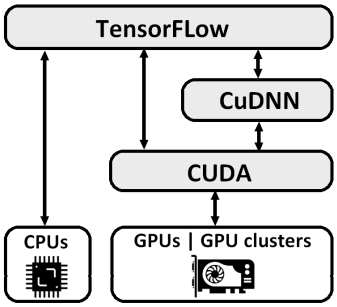



Figure 2.6: TensorFlow uses CUDA and cuDNN to control GPUs and boost DNNs.

and FC are fully-connected layers, where the last one computes the logits of different classes just before a softmax classifier. There are several Convolutional Networks architectures. The most common ones are:

- AlexNet. [3] was The first work that popularized Convolutional Networks in Computer Vision, it was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error),
- GoogLeNet. [9] Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, it uses Average Pooling instead of Fully Connected layers at the top of the ConvNet,
- ResNet. [31] It was demonstrated that above a certain number of stacked layers, the network performances start to decrease, ResNet solves this by using special skip connections (forcing the network to model $f(x) = h(x) - x$ rather than $h(x)$) with a heavy use of batch normalization [24] (i.e. normalizing the input layer by adjusting and scaling the activations). The architecture is also missing fully connected layers at the end of the network. ResNets are currently by far state-of-the-art Convolutional Neural Network models and are the default choice for using ConvNets in practice, We note that this is the backbone architecture our model is based on.

## 2.6 Deep Learning Software

Deep learning frameworks offer building blocks for designing, training and validating deep neural networks, through a high level programming interface. Widely used deep learning frameworks such as Caffe2, MXNet, PyTorch, TensorFlow and others rely on GPU-accelerated libraries such as cuDNN and NCCL to deliver high-performance multi-GPU accelerated training.

Nvidia's CUDA allows developers to use CUDA enabled GPUs for all sorts of computations. Nvidia's CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for DNNs. It provides optimized implementations of common DNN computations such as activation layers, normalization, forward and backward convolutions, and pooling as part of Nvidia's Deep Learning SDK.

In this work, we'll use TensorFlow [40], TensorFlow is a powerful open source software library for numerical computation, particularly well suited and fine-tuned for large-scale Machine Learning. Its basic principle is simple: first we define in Python a graph of computations to perform, and then TensorFlow takes that graph and runs it efficiently by using optimized C++ code and utilizing CUDA and cuDNN to control the GPU cards and accelerate computations (Figure 2.6).

## 2.7 Workflow

A typical workflow for applying a neural network in some context looks as follows:

**Data preparation.** First, we obtain a dataset. For the purposes of this work we have a set of pairs $(x, y)$ where $x$ is some input example and $y$ is a label (objects in the image with their respective bounding boxes and a set of triplets for the occuring interactions). We then split the dataset into two folds, a training and validation (80% and 20% respectively). We will use the training fold for optimizing the parameters with backpropagation and the validation fold for hyperparameter optimization and evaluation.

**Data preprocessing.** Preprocessing the data can help improve convergence of neural networks [60]. For images, common preprocessing techniques involve standardizing the data (subtracting the mean and dividing by the standard deviation individually for every input dimension of $x$). Data augmentation techniques can also be used, like horizontal filpping or rescaling the image to take into consideration a limited number of labels (e.g. only one interaction triplet in the image at the time).

**Architecture design.** Next we choose a family of architectures that we wish to explore. This amounts to designing the internals of the computation graph that makes up the function $f$.

**Hyperparameter optimization.** The optimization with stochastic gradient can be seen as an inner loop of the optimization, while hyperparameter optimization is the outer loop that determines good values of hyperparameters that are difficult or impossible to backpropagate into (such as the learning rate, or the number of units in the hidden layers). This process consists of sampling hyperparameters from some search range, optimizing the model, and evaluating the model on the validation fold. The final best model is the one that achieves the best validation performance.

**Evaluation.** Once we identify the best trained model (with the lowest validation loss), we evaluate the model a single time on an unseen data and report the performances.

# Chapter 3

# State-of-the-art

Given that in this work, our two main contributions are a new dataset for Human-Interactions and a one-stage model for Human-Interactions detection and recognition with two principal components, an object detection module and a human interactions module (Figure 3.1). This chapter presents the state-of-art of object detection, relationships and interactions recognition systems with on overview of the main datasets used in each one of these applications.

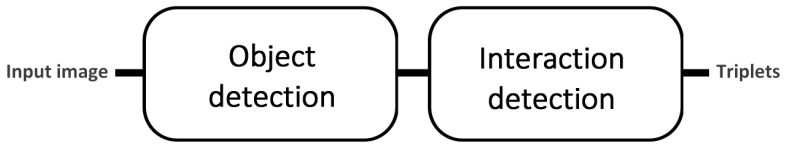



Figure 3.1: A high-level perspective on the Human-Interactions detection model to be presented later. It contains two modules, one for object detection taking an input image and outputing a list of objects with their respective labels and bounding boxes, the second module takes these detections as inputs and outputs the interactions taking place in the form of triplets (<*subject, verb, object*>).

## 3.1 Object Detection

Object Detection is about identifying and locating all known objects in a scene. With the progress that has been made in recent years on object detection due to the use of ConvNets, Modern object detectors based on these networks are now good enough to be deployed in consumer products (e.g. Google Photos) and some have been shown to be fast enough to be run on mobile devices. In this section we'll go through the important Object Detection concepts and present some of the modern object detectors.

### 3.1.1 A brief history

In the 1970's one of the first approches to face detection was using constellation models specifying the relative positions of each component to an other (nose, eyes, hair, etc.), and then a face is detected if each component is roughly in its correct relative position. In 1980's, with the AI winter, the computer vision research encountered some difficulties, the majority of the work in this period focused on the basics such as edge detection (e.g. Canny detector [8]) and on how to represent objects, in late 80's computer vision went from a Science to an Engineering branch with the appearance of application based methods such as CNNs [60] applied to zip codes recognition (unfortunately CNNs vanished shortly after). Fast forward to the 2000's where a lot of impressive advancements took place, like rapid object detection using boosted

cascade of simple features [57] (i.e. sliding Haar wavelets) that worked quite well on face detection in standard cases, the use of SIFT detector [37] for object instance recognition, but these detectors worked only with specific kind of objects in certain conditions and failed for others. The first repeatable method was using Histogram of oriented gradients [58] for pedestrians detection (i.e. by sliding SIFT detectors with local normalizing and an SVM classifier on top) but still had problems for objects not under normal conditions. Deformable parts [44] tried to solve this by using both HOG detectors and constellation models. With these types of feature based detectors, very good results were obtained in object detection datasets but for small error gains the complexity of the detectors pipeline increased greatly. CNNs came back once again after the results AlexNet demonstrated on image classification and were also applied to object detection giving human like performances and real time capabilities to modern object detectors.

### 3.1.2 Important Object Detection Concepts

In general, modern object detectors use similar concepts with minor adjusments to obtain higher accuracy and greater speed. In this section we go through the important concepts that are shared between object detection systems:

**Bounding box proposal** (region of interest, region proposal, box proposal). A rectangular region of the input image that potentially contains an object inside. These proposals can be generated by some heuristics search: objectness, selective search, or by a region proposal network (RPN). A bounding box can be represented as a 4-element vector, either storing its two corner coordinates $(x_0, y_0, x_1, y_1)$, or (more common) storing its center location and its width and height $(x, y, w, h)$. A *bounding box* is usually accompanied by a confidence score of how likely the box contains an object. The difference between two bounding boxes is usually measured by the L2 distance of their vector representations. $w$ and $h$ can be log-transformed before the distance calculation.


Figure 3.2: An example of bounding box regression, the detected box (red) is misplaced and does not match the correct one (blue), we hope that the network can recognise that the object detected (dog) is missing some of its parts and outputs (i.e. the regressor) an offset $\triangle(x, y, w, h)$ to correct the predicted box.

**Bounding box regression** (bounding box refinement). By looking at an input region, we can infer the bounding box that better fit the object inside, even if the object is only partly visible (see Figure 3.2). Therefore, one regressor can be trained to look at an input region and predict the offset $\triangle(x, y, w, h)$ between the input region box and the ground truth box. If we have one regressor for each object class, it is called class-specific regression, otherwise, it is called class-agnostic (one regressor for all classes). A bounding box regressor is often accompanied by a bounding box classifier (confidence scorer) to estimate the confidence of object existence in the box. The classifier can also be class-specific or class-agnostic. Without defining prior boxes, the input region box plays the role of a prior box.

**Intersection over Union** (IoU, Jaccard similarity). A metric that measures the similarity between two bounding boxes, i.e. $IoU$ = their overlapping area / their union area.

**Non Maximum Suppression** (NMS). A common algorithm to merge overlapping bounding boxes (proposals or detections). Any bounding box that significantly overlaps with a higher-confident bounding box is suppressed (removed).

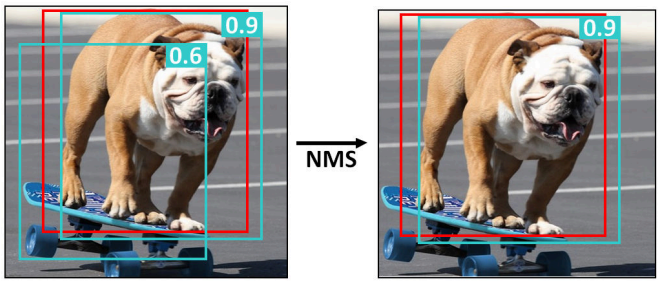



Figure 3.3: **Left**, two bounding boxes are detected (blue), both have $IoU > IoU_{threshold}$ (e.g. $IoU_{threshold} = 0.5$) with the ground truth box (red). **Right**, after applying NMS, only the box with the highest confident score ($0.9 > 0.6$) is preserved.

**Anchor box** (default box, prior box). Instead of using the input region as the only prior box, we can train multiple bounding box regressors, each look at the same input region but has a different prior box and learns to predict the offset between its own prior box and the ground truth box. This way, regressors with different prior boxes can learn to predict bounding boxes with different properties (aspect ratio, scale, locations). Prior boxes can be predefined relatively to the input region, or learned by clustering. An appropriate box matching strategy is crucial to make the training converge.

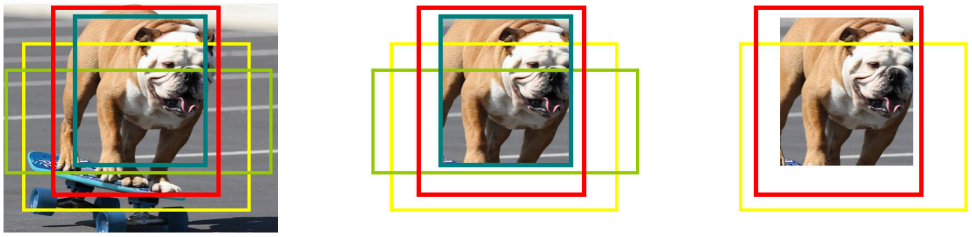


Figure 3.4: **Left**, One region proposal (blue) with 2 anchor boxes (green and yellow) and one ground truth box (red). **Middle**, The 2 bounding box regressors only see the input region and try to infer the ground truth box from their prior boxes. **Right**, In Multibox strategy, the ground truth box is matched with the anchor box with highest IoU.

**Box Matching Strategy.** We cannot expect a bounding box regressor to be able to predict a bounding box of an object that is too far away from its input region or its prior box. Therefore, we need a box matching strategy to decide which prior box is matched with a ground truth box. Each match is a training example for regressing. Possible strategies: 1- Matching each ground truth box with one prior box with highest $IoU$ (i.e. Multibox strategy, see Figure 3.4); 2- Matching a prior box with any ground truth with $IoU$ higher than $IoU_{threshold}$ (e.g. 0.5).

**Hard negative example mining.** For each prior box, there is a bounding box classifier that estimates the likelihood of having an object inside. After box matching, all matched prior boxes are positive examples (i.e. containing an object) for the classifier. All other prior boxes are negatives (i.e. background). If we used all of these hard negative examples, there would be a significant imbalance between the positives and negatives. One possible solution is to randomly pick negative examples, or pick the ones that

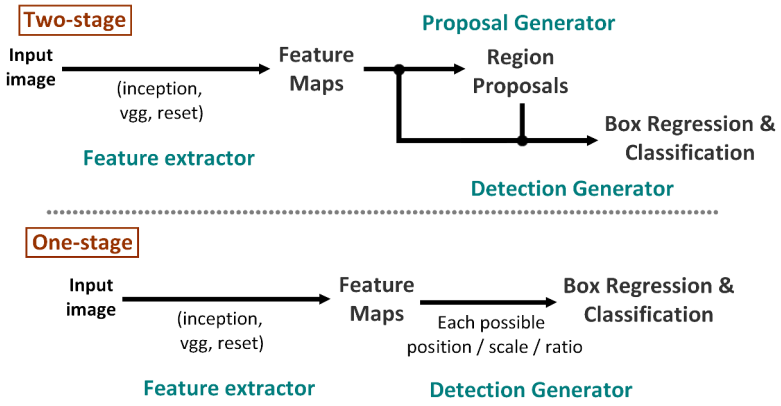



Figure 3.5: High level diagrams of the two detection paradigms. **Top**. Where we propose a set of condidates (proposals) either by using an external algorithm or a RPN (i.e. from a set of anchors detect a set positives and negatives) and then classify them and regress their boxes. **Bottom**. Where the proposals are predefined (i.e. anchors) and the classification and regression are done directly.

the classifier makes the most serious error, so that the ratio between the negatives and positives is at roughly 3:1.

**Evaluation metrics.** The metric used for object detection is mean Average Precision (mAP). It is the average of the maximum precisions at different recall values. Given positives (i.e. boxes with IoU $\geq$ IoU$_{threshold}$) and negatives (i.e. IoU $<$ IoU$_{threshold}$), we measure Precision at various equally spaced Recall levels (e.g. $0, 0.1, ..., 1$) by varying the ranks (the number of boxes we consider) and using interpolation if needed (e.g. Equation 3.1, $r$ for recall, $p$ for precision and $p_{inter}$ for the interpolated precision), the Average Precision (AP) [12] is the average of all these Precision measures (e.g. Equation 3.2)

$$p_{inter}(r) = \max_{\bar{r} \geq r} p(\bar{r}) \tag{3.1}$$

$$AP = \frac{1}{11} \sum_{r \in [0,..,1]} p_{inter}(r) \tag{3.2}$$

mAP [56] is simply the mean of the AP values calculated for different IoU thresholds (e.g. [0.5, ..., 0.95]).

### 3.1.3 The two approaches

Given that the objects can appear in the image at any position with any relative size, the naive brute-force approach is to test every possible position using a sliding window with every possible aspect ratio/scale, giving us an exponential number of combinaison. There is two competing ways to solve this problem:

- **Two-stage detectors:** The first stage is the Region Proposal, outputing the fixed number of boxes where an object might be present (e.g. 2000 proposals) with a high recall (i.e. if there is an object, it will likely be proposed) by using external proposal generator (e.g. blob detectors like Selective search [25], where adjacent regions with similar properties such as color and texture are grouped together), these proposals are then sent down the pipeline for object classification and bounding-box regression outputing a vector of length $6 \times N_{proposals}$ (For each proposal we have the class label, the confidence score and the offset $\triangle(x, y, w, h)$).

- **One-stage detectors:** Instead of first predicting region proposals, these type of detectors use predefined positions, scales and ratios and treat object detection as a simple regression problem by taking an input image and learning the class probabilities and bounding box coordinates.

The first approach has region proposals that fit the objects better than the other grid-like candidate windows but is orders of magnitude slower. The second approach takes advantage of the convolution operation to quickly regress and classify objects in sliding-windows fashion but lacks in terms of accuracy.

With the introduction of Anchor boxes and Region Proposal Network (RPN) [6, 7, 10, 49] . All state-of-the-art methods now has a set of anchor boxes (generated based on a set of sliding windows or by clustering ground-truth boxes) from which bounding box regressors are trained to propose regions that better fit the object inside. Similarly, the new competition is between a one-stage **direct classification** (i.e. based on the anchor boxes do the classification and regression directely) and a two-stage **refined classification**, i.e. instead of external proposal generator, first use a RPN to do a first round of classification and regression of the anchor boxes, only detecting if there is an object or not. Then in the second round classify the proposals into specific object classes together with regressing the coordinates.

### 3.1.4 Modern Object Detectors

Here, we present some of the most used object detectors. Please refer to [27] for more in depth look and apples-to-apples comparison of these detectors and other ones not mentioned in this work:

**R-CNN & Fast R-CNN** (Two-Stage). R-CNN [45] is a natural combination of heuristic region proposal method and ConvNet feature extractor. From an input image, $\sim$ 2000 bounding box proposals are generated using selective search [25]. Those proposed regions are cropped and warped to a fixed-size $227 \times 227$ image. AlexNet is then used to extract 4096 features (i.e. from layer FC7) for each warped image. An SVM model is then trained to classify the object in the warped image using its 4096 features. Multiple class-specific bounding box regressors are also trained to refine the bounding box proposal using the 4096 extracted features. With Fast R-CNN [18] The proposals crops are not taken directly from the image and re-run through the feature extractor, which would be duplicated computation. Instead the crops are taken from the feature maps (e.g., CONV5 in VGG-16 [50]). However there is part of the computation that must be run once per proposal, and thus the running time depends on the number of proposals by the selective search.

**Faster R-CNN** [49] (Two-Stage, refined classification). First, images are processed by a feature extractor (e.g. VGG-16), and the features at some selected intermediate level (e.g., CONV5) are used by region proposal network (RPN) to predict class agnostic box proposals using a grid of anchors tiled in space, scale and aspect ratio. Second, these (typically 300) box proposals are used to crop features from the same intermediate feature map which are subsequently fed to the remainder of the feature extractor (e.g., FC6 followed by FC7) in order to predict a class and class-specific box refinement for each proposal using the proposals generated from the RPN as anchors.

**YOLO** [29] (One-Stage, direct classification). Adds a softmax layer, parrallel to the box regressor and box classifier layer in the RPN, to directly predicts the object class. In addition, instead of clustering ground truth box locations to get the prior boxes, YOLO divides the input image into a $7 \times 7$ grid where

each grid cell is an anchor box. The grid cell is also used for box matching: if the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. The box regressor, confidence scorer, and object classifier look at features extracted from the whole image.

**RetinaNet** [55] (One-Stage, direct classification). RetinaNet draws on a variety of recent ideas from [54, 10, 59, 49]. Given that it is efficient (i.e. a one-stage detector) and accurate (i.e. the use of focal loss). It was the object detector of choice in this work. We will go through the main elements making RetinaNet acheives both high speed and accuracy:

**1- Feature Pyramid Network.** Detecting objects at vastly different scales is a fundamental challenge in object detection, ConvNets are capable of representing higher-level semantics (e.g. high level features like faces), and are also more robust to variance in scale than traditional object detectors like DPM [44] and thus facilitate recognition from features computed on a single input scale, but even then, when doing predictions (object classes and bounding boxes) based on the last computer feature maps, and due to the pooling layers (i.e. reducing the spacial dimensions) a lot of small objects are not being detected (Figure 3.6 **Left**).

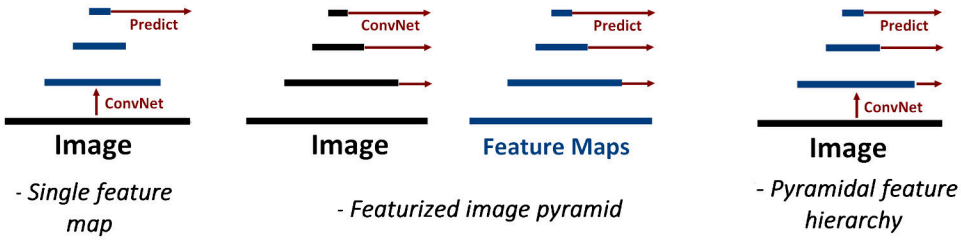



Figure 3.6: Detecting objects at different scales. **Left**. Only do the prediction from the last feature maps containing high-level semantics but low resolution due to the repeated pooling. Fast but small objects are difficult to detect. **Middle**. Resize the image and forward it each time into a ConvNet (e.g. ResNet, VGG) and do the predictions seperately for each one. Slow but yields good results. **Right**. Use the pyramidal shape of the ConvNet to detect object at different scales. Fast but when doing predictions based on intermediate level, the semantic level is low. Thus yielding average results.

A naive approach to deal with the scale invatiance is to resize the image to multiple scales (i.e. building an image pyramid) and do the detections separately for each level, ending up with good results but a very slow run times (Figure 3.6 **Middle**). The Single Shot Detector (SSD) [59] is one of the first attempts at using ConvNet's pyramidal feature hierarchy as if it were a featurized image pyramid, it reuses the multi-scale feature maps from different layers computed in the forward pass and thus come free of cost in terms of computation, the prediction are than done seperately for each feature level (e.g. starting from CONV4_3 of VGG [50] or CONV2 of ResNet [31]). When starting from an intermediate level, the detector is avoiding low-level feature (e.g. edges, lines, etc.) but only using mid-level features (e.g. a combination of horizental lines). Thus it misses the opportunity to reuse the higher-resolution maps of the feature hierarchy (i.e. Figure 3.6 **Right**).

**FPN** (Figure 3.7). A similar idea to the one used in SSD detector, leveraging the pyramidal shape of a ConvNet's feature hierarchy while creating a feature pyramid that has strong semantics at all scales. This is achieved by using an architecture that combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections. Starting from the last feature map, using a $1 \times 1$ convolution to adjust the depth (i.e. all the levels are converted

into 256 feature maps), the top feature map is 2× upsampled (i.e. nearest neighbor upsampling) to adjust the spatial resolutions and then merged with the bottom feature, this process is done for last four residual blocks $\{C2; C3; C4; C5\}$ in ResNet. The result is a feature pyramid that has rich semantics at all levels and is built quickly from a single input image scale.

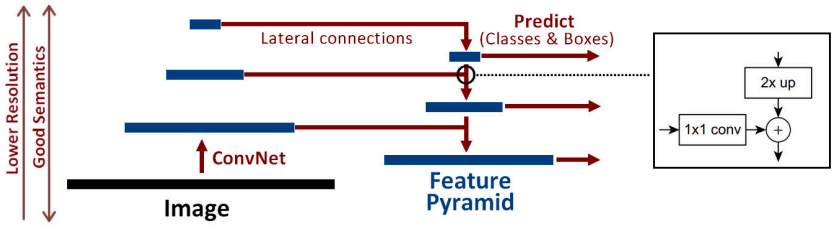



Figure 3.7: An illustration of the lateral connection and the top-down pathway used in FPN, merged by addition. Resulting in a feature pyramid of varying resolution but with strong semantics at all levels. Thus giving us the ability to detect objects at various scales.

**2- Focal Loss** One of the main difficulties in object detection is class imablance, in an image there is an almost infinite number of background instances and just a few number of objects. The loss induced by the background instances can dominate total loss. Resulting in an inefficient training as most locations are easy negatives that contribute no useful learning signal and lead to degenerate models.

Two-stage detectors address class imbalance through two mechanisms: (1) An object proposal mechanism [25, 6] that reduces the nearly infinite set of possible object locations down to $\sim$ 2000. (2) The selected proposals for the second stage are not random, but are either true object locations, or hard negatives. With a 1:3 ratio of positive to negative examples.

In a one-stage scenario, without a first stage to eliminate the easy examples, there is an extreme imbalance between foreground and background classes during training (e.g., 1:1000). RetinaNet solves this problem by introducing Focal Loss (i.e. a modified version of the cross entropy loss):

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{3.3}$$

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases} \tag{3.4}$$

In Equation 3.4, $y \in \{\pm 1\}$ specifies the ground-truth class and $p \in [0, 1]$ is the model's estimated probability for the class with label $y = 1$.

In Equation 3.3. The weight $\alpha$ balances the importance of positive/negative examples by giving a larger weight to the rare classes. And a modulating factor $(1 - p_t)^{\gamma}$ with tunable focusing parameter $\gamma \geq 0$ to adjust the rate at which easy examples are downweighted (e.g. $\gamma = 2$ is used in the original paper [55]), it reduces the loss contribution from easy examples and extends the range in which an example receives low loss (e.g. with $\gamma = 2$, an example correctely classified with $p_t = 0.9$ would have 100× lower loss compared with the normal cross entropy. While in turn increasing the importance of correcting misclassified examples with $p_t \leq 0.5$).

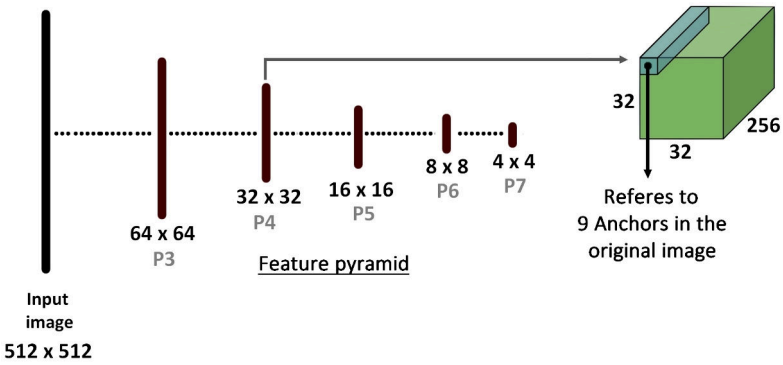



Figure 3.8: The anchors defined in RetinaNet. For each level of the pyramid, the number of anchors defined in the original image is equal to the the spatial dimension in that level $\times 9$ (i.e. 3 ratios & 3 scales), E.g. for P3 and an input image of size $512 \times 512$, the spatial dimension are $64 \times 64$, thus, at each one of the $64 \times 64$ locations (i.e. at various strides) in the original image we have 9 anchors with different scales and ratios. Resulting in $64 \times 64 \times 9$ anchors in total.

**3 - Anchors** The way anchors are defined in RetinaNet is based on the pyramid levels (see Figure 3.8). From P3 to P7, for each level we have number of defined anchors in the original image, where in each location we have 9 anchors, e.g. 3 ratios {1:2; 1:1, 2:1} and for denser detection, we have 3 scales $2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}$. For instance, if the input image is of size $512 \times 512$, we will have a total number of anchors equal to $9 \times (64^2 + 32^2 + 16^2 + 8^2 + 4^2) = 49104$. For each level there a varying strides in the original image (e.g. In Google's implementation[1] they use a stride starting from pixels $(\frac{2^{level}}{2}, \frac{2^{level}}{2})$), e.g. for P3 we will have a Stride of $2^{level}$, thus we'll have at each position in the image starting from the pixel $(4, 4)$ with strides of 8 (e.g. $(4, 8)$, $(8, 4)$, $(8, 8)$, ...) 9 anchors. The base scale is then calculated for each scale (e.g. $base_{scale} = Anchor_{sclae} \times Stride \times Scale$, With $Anchor_{sclae} = 4$) and then adjusted for the defined ratios. The highest the level the bigger the base scale.

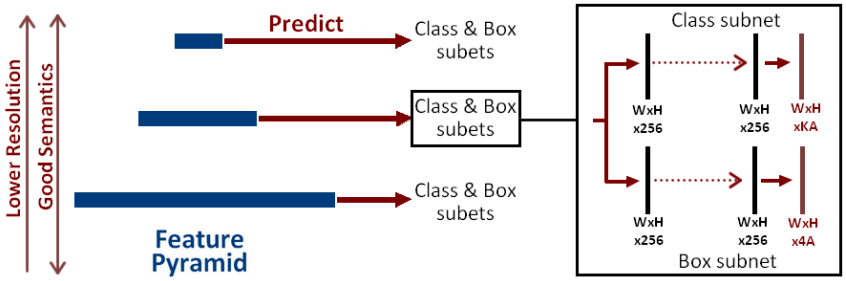



Figure 3.9: RetinaNet network architecture uses a Feature Pyramid Network (FPN) (generated from a feedforward ResNet architecture). To this Feature Pyramid RetinaNet attaches two subnets, one for classifying anchor boxes and one for regressing from anchor boxes to ground-truth object boxes, a simple architechture compared to a one-stage detector, but with similar accuracy and higher speed.

**Predictions** At each level of the feature pyramid, two subneworks are attached. Each one contains four $3 \times 3$ consecutive convolutions and ending with a depth of either $KA$ or $4K$ (i.e. A anchors and K classes). Class subnet predicts the probability of object presence at each spatial position for each of the $A$ anchors (i.e. ratios $\times$ scales) and $K$ object classes (e.g. $A = 9$ and $K = 80$). Box subnet with identical design except that it terminates in $4A$ (i.e. 4 offsets $\triangle(x, y, w, h)$) linear outputs per spatial location.

[1] **Link** to Github repository

Each anchor is assigned a length $K$ one-hot vector of classification targets (i.e. a vector of $K$ elements, with all elements = 0 except the class to which the anchors is assigned to, we note that the backgound class has label = $-1$), where $K$ is the number of object classes, and a 4-vector of box regression targets. Anchors are assigned to ground-truth object boxes using an $IoU_{threshold} = 0.5$, and to background if their $IoU$ is in $[0, 0.4)$. As each anchor is assigned to at most one object box, we set the corresponding entry in its length K label vector to 1. If an anchor is unassigned, which may happen with overlap in [0.4, 0.5), it is ignored during training.

## 3.2 Relationships and Interactions

### 3.2.1 Motivation

Classical approaches study the task of action classification, either detecting the main action taking place at the image or video level or at best produce a bounding box around the person doing the action. Such an output is inadequate and a complete understanding can only come when we are able to associate objects in the scene to the different semantic roles.

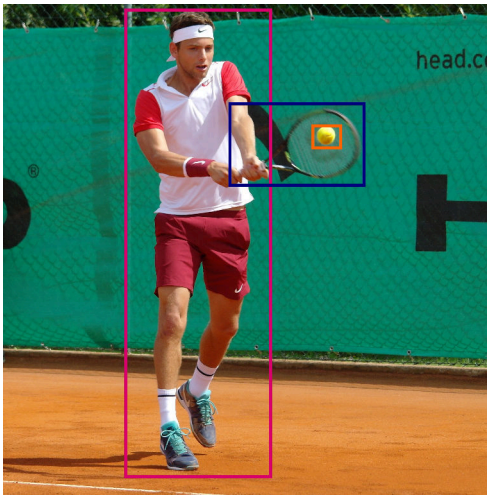


Figure 3.10: Going beyond a simple detection of the objects present in the image and the main action (e.g. hitting the ball) to being able to localize the agents, the instruments and the objects in various semantic roles associated with the action.

Figure 3.10 is an illustration of fine-grained interaction detection. Going beyond coarse activity labels such as "playing tennis", and being able to reason about sepecific actions such as "hitting" and detect the various semantic roles for this action namely: the subject (pink box), the instrument (blue box) and the object (orange box). Such an output can help us answer various questions about the image. It tells us more about the current state of the scene depicted in the image (association of objects in the image with each other and with actions happening in the image), helps us better predict the future (the ball will leave the image from the left edge of the image, the tennis racket will swing clockwise), help us to learn commonsense about the world (naive physics, that a tennis racket hitting a tennis racket impacts momentum), and in turn help us in understanding "activities" (a tennis game is an outdoor sport played in a field and involves hitting a round ball with a bat with a long handle attached to a round frame).

### 3.2.2 Relationships *vs.* Interactions

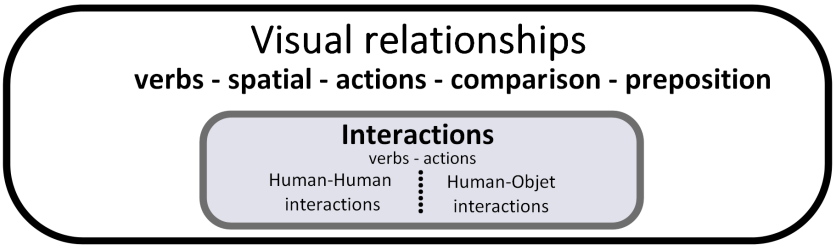



Figure 3.11: The set of relationships in a visual scene is more diverse and includes many types of possible associations (e.g. verbs, comparison, etc.). Interaction are a subset representing object directly associated with actions and verbs.

In this work we differentiate between relationships and interactions as follows. **Relationships** are derived from an open-world vocabulary referring to all the possible states an object can be in respect to an other one in an image/visual scene, such set of relationships include verbs (e.g. wear), spatial (e.g. next to), actions (e.g. ride), comparison (e.g. taller than), or a preposition phrase (e.g. with). Interaction are a subset of all the possible relationships, representing only the relations in which the object (i.e. a *subject*) is in direct interaction with an other *object* in the scene, either by a verb (e.g. person kicking a ball) or a direct action. We note that in some cases the *object* can not be present (e.g. *person* standing).

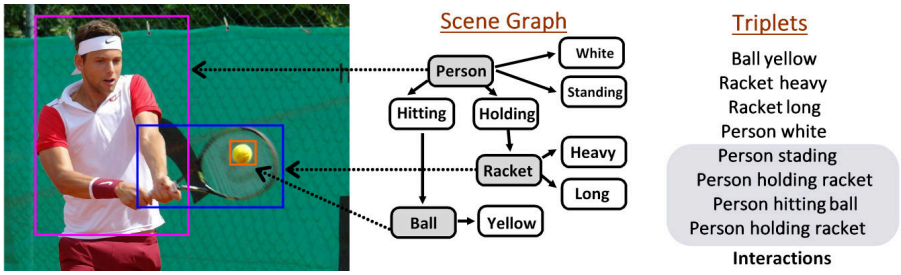



Figure 3.12: The representation of relationships and interactions. Relationships can be represented as Scene Graphs giving a richer semantic information about a visual scene with many direct and indirect possible associations of visual instances, and also as triplets. For the interactions, being constrained to representing only direct actions, are simply represented using triplets.

Relationships can either be represented as triplets *<object, relation, attribute>*, or by using scene graph with three types of nodes (see Figure 3.12); objects, attributes or relationships giving a richer representation and an easier way to find nested relations between objects. Interactions, representing only direct actions are generally represented with triplet *<subject, verb, object>*. We note in some cases the object might not be present.

**Focusing on the interactions**. In this work we aim to detect human-centric interactions (more specifically Human-Human interactions) using detectors capable of recognizing interactions in images with high precision, which is a requirement for practical applications. Detecting all relationships present in the image maybe tempting, but the models used for such tasks are more complex and less reliable. Further more in an open-world recognition setting, evaluating precision is not feasible (i.e. a fixed ground-truth is not available for a precision based metric to be used), resulting in recall-based evaluation, as in [5].

### 3.2.3 *State-of-the-art.* Relationships

With many possible applications such as caption generation, image search, visual question answering and image retrieval, research on visual relationship modeling [21, 5, 48, 38, 39, 4, 46] has attracted increasing attention. Provided that in this work, we are interested in detecting interactions, we will only take a glance at the current state of relationships recognition presenting some of the methods for detecting visual relations.

One of the principal challenges in visual relationships recognition is the number of possible combinations. Visual Genome [35], the main datasets for these types of applications, contains $33K$ unique object categories and $44K$ unique relationships. For detecting a set of triplets, the number of possible combinations is $N^2 \times K$ (i.e. $N$ objects and $K$ relations) which is in the order of $10^{14}$. Outputing a probability distribution over all possible triplets (i.e. using $N^2 \times K$ outputs) with a single end-to-end system is not applicable in this case. The other challenge is the non-uniform distribution of all triplet in the training set, some of the instances are frequent (e.g. cat on street), others are rare (e.g. dog ride skateboard), and yet, we would like to train a model to predict the correct triplets for the correct input regardless of its frequency in the training set. A supervised learning method is naturally prone to optimize the training targets that are based on a biased training set (i.e. prefering common examples over the rare ones).

**Visual Relationship Detection with Language Priors** In [5] they solve this problems by using two models, a visual appearance module for managing the $N^2 \times K$ conbinations and a language model for the rare instances. First an object detector such as R-CNN [45] detects all the objects in the image with a bounding box around them. The visual model then takes a pair of detected objects and their union and outputs a probability over all possible relationships (i.e. needing only $N + K$ outputs for $N$ objects and $K$ relationships), if it incorrectly labels the relationship (e.g. *person in horse*). They introduce a language module to aid the visual module, taking the individual word embeddings of a particular relationship triplet and outputs a probability of how likely that triplet will occur just based on the language embeding (e.g. outputing a low probability for *person in horse*). Together they give a probality of a relationship to occur based on both the bounding boxes and the language priors.

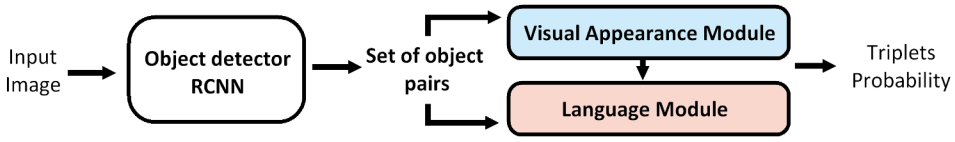



Figure 3.13: **Visual Relationship Detection**. Detecting relationships by first detecting the objects and a possible relationship using a visual model, and then using language priors to confirm the detected relationship.

**Detecting Visual Relationships With Deep Relational Networks** [4] Using a similar idea to [5] with two stages, a first stage for treating each component separately and detecting the objects first, and a second one for joint recognition. After detecting the object and a pair filtering to maintain only the important candidates. The second stage input is the union of the two object bounding boxes features and their spatial positions in the original image to detect a possible relationship. And instead of using a language module they use a probabilistic module (i.e. a conditional random field) implemented as a neural net (i.e. a deep relational network). Thus yielding better results than [5] with possible end-to-end training.

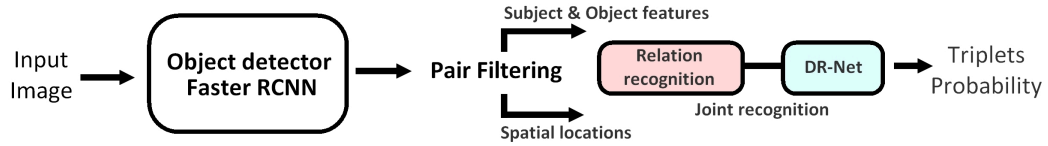



Figure 3.14: **Deep Relational Networks**. Using two models, one visual for object detection and pair filtering, and a probabilitic model implemented as a neural net for exploiting the statistical relations between the detected objects and the target relationship.

**Referring Relationships** One of the main problems with using two models as in [4, 5], is that it can be challenging to pinpoint whether errors made by these models occur from either the language or the visual components. In [46] they tackle a special case of referring relationships, taking as inputs both the image and the set of triplets, the goal is to detect the subjects and the objects in question alleviating the need to language model. This is done by using attention shift, starting from a possible *object* and *subject* positions and estimating the *subject* and *object* positions respectively. If the final position is the same as the starting position for the both (i.e. before and after the attention shift), the estimated positions of the *object* and the *subject* are likely to be correct.

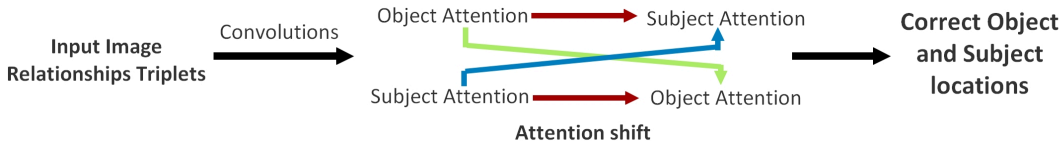



Figure 3.15: **Referring Relationships**. Taking as inputs both the image and the triplets, thus needing only one visual model, by using attention shift the correct locations of the object and subject are then predicted.

We note that there is other works that are interested in detecting more precise descriptions, e.g. [39] tackles the problem of Situation Recognition, with outputs in the form of *An AGENT VERB an ITEM from a SOURCE using a TOOL in a PLACE.*

### 3.2.4 *State-of-the-art.* Interactions

In Human interactions detection [1, 61]. The challenges of relationships detection are alleviated. The number of possible combinations can now be managed with $N^2 \times K$ outputs, given that the number of object categories and verbs is limited in comparison (e.g. $N = 80$ classes in COCO [56] and $K = 26$ verbs in VCOCO [21]). But similar challenges are present. Human actions are more fine-grained (e.g. walking, punching, dancing) than the actions of general subjects, and an individual person can simultaneously take multiple actions. These issues require a deeper understanding of human actions and the objects around them and in much richer ways than just the presence of the objects in the vicinity of a person in an image. Like relationships detection, accurate recognition of human-object interaction can also benefit numerous tasks in computer vision, such as action-specific image retrieval, caption generation, and visual question answering.

**Evaluation metrics.** Similarly to object detection (see Section 3.1.2), the evaluation metric used is Average Precision (AP) with slight modifications as defined in [21]. For the interaction, when detecting triplets <*subject, verb, object*>, there is two types of AP:

$\text{AP}_{agent}$ : Considering only the subject. Positives are when the IoU($subject$) > $\text{IoU}_{threshold}$ (i.e. The IoU between *subject* box and the ground truth > $\text{IoU}_{threshold}$) and the verb associated with the detected subject is correct. We then proceed to calculate the $AP$ exactly the same manner as in object detection.

$\text{AP}_{role}$ : In this case we consider both the subject and the object. Positives are when both the IoU($object$) > $\text{IoU}_{threshold}$ and IoU($subject$) > $\text{IoU}_{threshold}$ and the verb associated with both is the correct one. We note that there is two version of $\text{AP}_{role}$ as defined in [21]. $\text{AP}_{role1}$ when we have an individual actions (i.e. no object, e.g. *person standing*), the score of the object must be zero to have a correct positive. For $\text{AP}_{role2}$ the score of the object is not considered.

We present three main Human-Object interaction detection methods :

**Visual Semantic Role Labeling** In [21] they use two models in succession. The first model (i.e. Fast-RCNN object detector [18]) for detecting the person instances present in the images with their respective bounding boxes, and then classifies the detected people into different action categories (i.e. a multi label classification given that a single person can be participating in various actions simultaneously). After detecting the subject, the second model comes into action to detect the object with a regression in the coordinate frame of the detected agent (i.e. similar to object detection when regressing the detected boxes to the ground truth ones). This method is based only on the human features to detect the interaction, missing the opportunity to use the features of the objects present in the scene and their relative spatial positions.

**Learning to Detect Human-Object Interactions** Contrary to [21], [63] propose a new multi-stream DNN-based framework that exploits features not only from human boxes, but also the objects and their spatial relations. This is done in two stages. First, by generating human proposal where each proposal is a human-object pair using an object detector (i.e. Faster-RCNN [49]) to detect all objects in the image, and then pairing each detected object and human to get one proposal. Second, these proposals are classified using the multi-stream network, taking as inputs the features extracted from both the human and object bounding boxes and their absolute spatial positions and outputing the interaction (i.e. a label of the verb) taking place. The problem with this approach is the $O(n^2)$ complexity of multi-stream network (i.e. considering all human-object pairs even when no interaction is taking place between them), hence, negatively impacting the inference time.

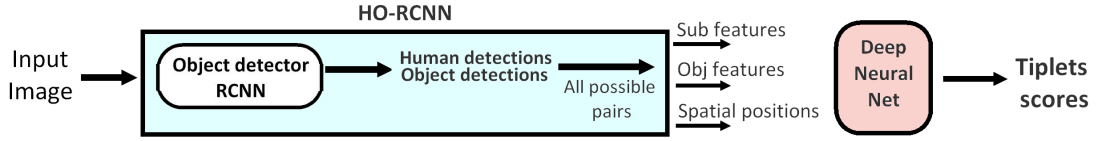



Figure 3.16: Detecting human interactions by taking each pair of object-human boxes as a possible combination and outputing the scores of the interactions taking place.

**Detecting Human-Object Interactions** In trying to solve the $O(n^2)$ complexity in [63] and using both the features from object and human instances as opposed to [21]. And based on a simple hypothesis,

that the appearance of a person, their pose, clothing and action is a powerful cue for localizing the objects they are interacting with. [17] propose InteractNet. A model containing 3 branches. **(1)** An *Object Detection* branch indentical to Faster-RCNN [49] outputing the object classes and bounding boxes present in the image. **(2)** A *Human-Centric* branch taking as inputs the detected human-boxes, and outputing action labels (i.e. the actions of which the agent in the human-box is the *subject*) and target localization as a density over possible locations for predicting the *object* location (i.e. a Gaussian function $g^a_{h,o}$, representing a density over the location of object box $b_o$ associated with an action $a$ and a human box $b_h$). **(3)** An *Interaction* branch for using the features form the detected object ($b_o$), the features of the human box ($b_h$) and the ouputs of the Human-Centric branch (i.e. action labels and the density locations) to test the compatibility of the predicted position with the actual position of the *object*.

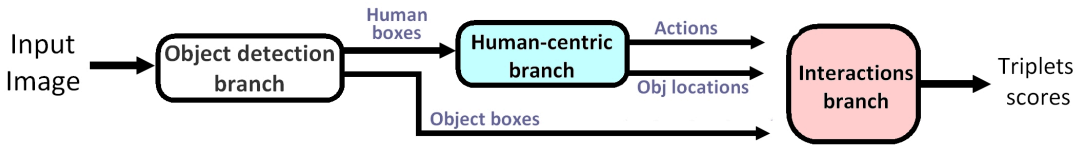



Figure 3.17: **InteractNet**, Detecting human interactions using a human centric approach, with three branches, one for object detection, one for action recognition and object location prediction based on the human boxes $b_h$, and finally the interaction branch outputs the correct triplets.

**InteractNet** acheives greater speed (i.e. inference time $\approx 135ms$) due to the fact that only the third branch is of complexity $O(n^2)$, and it is only used for fast-checking (i.e. with negligible time compared to the time spent computing the full-image shared convolutional feature maps). And better accuracy with the usage of both the features of the human box and the object box.

## 3.3 Datasets

In this section we will present the main datasets used for various computer vision recognition and detection tasks. We note that generally, the annotation format changes from dataset to an other with minor similarities. Please refer to Appendix B for an overview of the most common annotation formats.

### 3.3.1 Object detection

The two main datasets used for object detection are PASCAL VOC[2] [12] (i.e. two versions, VOC 2007 and VOC 2012) and COCO[3] [56]. PASCAL VOC is currently outdated and all of the state-of-the-art methods use COCO dataset due to the lack of diversity and the results obtained are not reflective of the model's true performances. See Table 3.1 for a brief comparison. We note that there are many dataset for specific tasks, e.g. Cars Dataset [28], Berkeley's Deep Drive [15] for road object detection and CORe50 [36] for continuous learning and object recognition.

| Datasets | Size | Classes | Detection | Segmentation | Key points | Actions | Captions |
|---|---|---|---|---|---|---|---|
| Pascal Voc [12] | 17.5K images | 20 | ✓ | ✓ | ✗ | ✓ | ✗ |
| COCO [56] | 123K images[4] | 80 | ✓ | ✓ | ✓ | ✗ | ✓ |

Table 3.1: A comparison of the two main datasets used in object detection tasks.

[2] **Link** to PASCAL-VOC dataset
[3] **Link** to COCO dataset

### 3.3.2 Relationships

In recent years, many datasets targeted towards relationships recognition have appeared, Table 3.2 present some of them.

| Datasets | Size | Relations | Scene Graphs | Triplets | Q&A[5] |
|---|---|---|---|---|---|
| Visual Genome [35] | 110K images | 30K | ✓ | ✓ | ✓ |
| VRD [5] | 5K images | 70 | ✓ | ✓ | ✗ |
| imSitu [39] | 125K images | 504 | ✗ | ✓ | ✗ |
| CLEVR [30] | 70K images | - | ✗ | ✗ | ✓ |

Table 3.2: Datasets for relationships recognition.

### 3.3.3 Human-Object interactions

For Human-Object interactions, i.e. where the *subject* is a person instance and the *object* is an object instance, resulting in triplets in the form of *<person, verb, object>*. The two main datasets are VCOCO [21] [6] and HICO [64] [7], HICO-DET [63] is the newer version of HICO with bounding box annotations of the *subjects* and *objects*.

| Datasets | Size | Object categories | Verbs | Triplets | Boxes Annotation of S&O[8] |
|---|---|---|---|---|---|
| HICO [64] | 47K images | 80 | 117 | ✗ | ✗ |
| HICO-DET [63] | 47K images | 80 | 117 | ✓ | ✓ |
| VCOCO [21] | 10K images | 80 | 27 | ✓ | ✓ |

Table 3.3: Datasets for Human-Object interactions.

HICO-DET is much richer in terms of H-O interactions, but VCOCO benefits from the different types of annotations used in COCO dataset (i.e. VCOCO is a subset of COCO).

### 3.3.4 Human-Human interactions

In Human-Human interactions, the *object* is also a person instance, i.e. we have triplets in the form of *<person, verb, person>*. Table 3.4 contains the main H-H interactions datasets.

| Datasets | Type | Size | Verbs | Triplets | Boxes of S&O |
|---|---|---|---|---|---|
| ShakeFive2 [14] | Videos | 153 | 8 | ✗ | ✗ |
| Human Interaction Images [13] | Images | 1972 | 9 | ✗ | ✗ |
| K3HI [52] | 3D coordinates | 312 | 8 | ✗ | ✗ |
| SBU Kinect Interaction [65] | Videos | 171 | 8 | ✗ | ✗ |
| BIT-Interaction [34] | Videos | 400 | 7 | ✗ | ✗ |
| UT-Interaction [47] | Videos | 60 | 6 | ✗ | ✗ |
| TV Human Interaction [43] | Videos | 350 | 5 | ✗ | ✗ |

Table 3.4: Datasets for Human-Human interactions.

[4] The total count of the images is $\sim$ 330K, but only $\sim$ 123K are labeled.
[5] Q&A: Question Answering.
[6] **Link** to VCOCO Github repository
[7] **Link** to HICO dataset
[8] S&O: Subject and object.

As it is obvious from Table 3.4, the Human-Human interactions datasets, contrary to the Human-object interactions datasets, lack diversity (i.e. limited number of verbs and images/videos), the right type of data (i.e. only one dataset contains images, the majority of which are of low quality) and more importantly, they don't provide the annotations of objects and interactions (i.e. the datasets do not contain any type of annotations, only labels for each image indicating the main action in it). Thus, we decided to create a Human-Human interactions dataset on par with the state-of-the-art datasets.

# Chapter 4

# Dataset creation

Given the limited number of examples, the lack of diversity and the provided annotations for H-H interactions as showed in Section 3.3.4. We decided to create a new dataset for these types of interactions. We set a target of **25 verbs**, 80 **Object categories** (i.e. the 80 categories used in COCO dataset [56]) and $\sim 5000$ **images**, while providing both **triplet** and **bounding box** annotations.

Our criteria for choosing the set of verbs we'd like to study is their simplicity, i.e. similary to VCOCO [21], we choose verbs that a 5 years old child can identify, the list of verbs is:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| arguing | bending | choking | dancing | feeding | fistbump | giving | saluting_waving | |
| handshake | helping | highfive | holding | hugging | interviewing | kicking | punching_boxing | |
| nursing | patting | playing | pulling_pushing | kissing | haircut | talking | throwing | thumbsup |

Table 4.1: List of chosen verbs for our dataset.

We note that similar verbs, e.g *Punching* & *Boxing*, *Saluting* & *Waving* and *Pulling* & *Pushing* are represented as a single verb in our dataset for simplicity, and given their semantic similarity.

To add some complexity to the dataset we choose to add some verbs that aren't representing direct interactions between the *subject* and the *object*, e.g. *Talking, Playing* & *Throwing*, detecting these verbs can be challenging, but we hope that the models to be used can infer these interactions from intermediate representations (e.g. objects and poses, two persons *playing* are both holding *controllers* and facing the same *direction*).

Some verbs refer to actions that are individually performed in most cases (i.e. no *object* instances), in our case the main verb containing these instances is *Dancing*, but can also be the case in other occurrences.

## 4.1 Collecting images

To populate our dataset, we choose three sources; **(1)** H-H interactions datasets, **(2)** Object detection datasets and **(3)** The web for completeness.

### 4.1.1 From H-H Interactions Datasets

To take advantage of the available datasets. We evidently decided to combine the existing Human-Human interactions datasets, for those that only contain videos, we took one still-image per interaction per video, the timing of the frames to be extracted are chosen to be representative of the action taking place and

the interaction must be easily deduced from them. Some frames may contain multiple interactions but the same extraction rules are applied.

Table 4.2 shows the extracted proportions from each dataset.

| **Datasets** | **Extracted Images or Videos** | **Extracted Verbs** |
|---|---|---|
| ShakeFive2 [14] | 153/153 | 6/8 |
| Human Interaction Images [13] | 1038/1972 | 7/9 |
| SBU Kinect Interaction [65] | 170/171 | 8/8 |
| BIT-Interaction [34] | 329/400 | 7/7 |
| UT-Interaction [47] | 60/60 | 6/6 |
| TV Human Interaction [43] | 151/350 | 4/5 |

Table 4.2: Extracting images from existing H-H interactions datasets.

We note that from the same video we may be able to extract multiple images (the case with UT-Interaction dataset).

### 4.1.2 From Object Detection Datasets

Given that we want to construct a dataset with complete annotations (bounding boxes and triplets), and to reduce the tedious effort to be spent later in the annotation process, we collect a number of images from pre-annotated datasets for object detection (i.e. from COCO and PASCAL-VOC) that already contain all the bounding boxes. Our task is to only collect the images in which the actions performed refer to one of the verbs we are interested in (Table 4.1).

A pre-pocessing step was first performed, extracting only the images where at least two instances of *person* appear (i.e. using the provided object annotations, *json* files in COCO and *xml* files in PASCAL-VOC). Reducing the images to be manually checked by 72% for PASCAL VOC 2012 (from 17.5K to 4.6K) and by 66% for COCO 2017 (from 123K to 40K).

The remaining images ($\sim$ 44.6K) are then manually checked for the presence of H-H interactions. The resulting extracted images from these datasets are shown is Table 4.3.

| **Datasets** | **Extracted Images** | **List of Extracted Verbs(# Images containing the verb)** |
|---|---|---|
| PASCAL | 290 | dancing(2) feeding(10) giving(5) handshake(1) helping(7) holding(216) hugging(4) kissing(3) playing(7) saluting_waving(1) talking(33) thumbsup(1) |
| COCO | 1766 | arguing(2) choking(1) dancing(1) feeding(45) fistbump(4) giving(52) haircut(44) handshake(62) helping(95) highfive(6) holding(649) hugging(21) interviewing(8) kicking(5) kissing(20) nursing(4) patting(1) playing(564) pulling_pushing(8) punching_boxing(1) talking(139) throwing(32) thumbsup(2) |

Table 4.3: Extracting images from existing object detection datasets.

### 4.1.3 From The Web

After extracting 3872 images from existing datasets, and having some verb instances without any examples, the next step was completing our dataset from the web, for this we used three ways:

**- Google Search:** Using A Python script to search keywords/key-phrases on Google Images and download the returned images, Google by default sets a limit of 100 image calls when interfacing with the Chrome browser, so to be able to exceed the download limit two libraries are used. *Selenium* (i.e. a portable software-testing framework for issuing tests and running them against most modern web browsers) along with *Chromedriver* (i.e. an executable that Selenium WebDriver uses to control Chrome). After choosing the right keyworks, prefixes, sufixes and specifying images with the rights to reuse them (i.e. One of the option of Google Image Search). We extracted around 2000 images per remaining verbs with various keywords (e.g. two person arguing, people having an argument, a dispute, etc.) and these images are filtered afterwords.

**- Flickr:** Flickr is known in the computer vision comunity for having good quality, diverse images with the rights to reuse them. Thus we wrote a simple web crawler using the REST Python API for downloading the images by inputting the correct URLs, i.e. The URLs are obtained doing manuel search in the website using different keywords and are then fed to the script as inputs to download the images. Then we manually remove bad examples.

**- Bing:** Finally, and to get as close to a uniform distribution in terms of the verb instances as possible. We used Microsoft's computer vision API[1]. An easy to use tool for doing various computer vision manipulations (e.g. reading handwritten text from images). In our case, we used it for downloading images from Bind Search Engine using custom keywords.

Table 4.4 shows the results.

| **Datasets** | **Extracted Images** | **List of Extracted Verbs(# Images containing the verb)** |
|---|---|---|
| Google Image Search | 1045 | arguing(7) bending(8) choking(114) dancing(91) feeding(8) fistbump(47) giving(5) haircut(51) handshake(47) helping(1) holding(24) hugging(93) interviewing(30) kicking(41) kissing(70) nursing(97) patting(1) punching_boxing(139) saluting_waving(52) talking(49) throwing(12) thumbsup(58) |
| Flickr | 206 | arguing(24) bending(7) choking(7) dancing(10) feeding(10) fistbump(11) haircut(15) handshake(2) holding(6) hugging(15) interviewing(33) kicking(7) kissing(12) nursing(8) punching_boxing(10) saluting_waving(7) talking(17) throwing(1) thumbsup(4) |
| Bing | 155 | arguing(40) bending(5) dancing(17) feeding(9) fistbump(9) giving(19) handshake(3) holding(8) patting(4) pulling_pushing(8) punching_boxing(14) saluting_waving(9) talking(1) throwing(9) |

Table 4.4: Completing our dataset with images extracted from the web.

We note that in both Tables 4.4 and 4.3, the numbers shown per verbs refer to the number of images of which the action present refer to the verb in question, not the number of interaction (i.e. for one image, we can have multiple interactions of the same type, thus the same verb).

### 4.1.4 Final Corrections

After collecting images of various sources, deleting all the bad instances, the final count is **5417 images**. These images have different fomats (e.g. `png`, `jpg`, `bmp`, etc.), different naming standards and different image qualities (e.g. the size of some images are $> 8Mb$). To normalize our dataset the following steps were applied:

[1] **Link** to MS's Computer Vision API

**- Image Compression.** All images with size $> 1Mb$ were compressed using lossless compression from the original format. This is done using `ImageMagick`.

**- Image Conversion.** All the images were converted to `jpg` due to its rich colors range (24-bits), being widely and accepted as the default image format. This is also done using `ImageMagick`.

**- Standardizing the dataset.** Before assigning the new `IDs` to the dataset images. A metadata file was created containing the original datasets information, i.e. For all the images the name of the source datasets were saved, for COCO and PASCAL-VOC the original `IDs` were conserved for each image and for the images extracted from the Web the original URLs and the websites domains were added to the metadata. For more details please refer to Appedix B. New `IDs` are then assigned to each image ranging from 0 to 50000 (i.e. in the same manner as COCO) and are then renamed depending on the obtained `IDs` (i.e. with 6 `zfill`, an image with `ID` 1 is then named 000001.jpg).

## 4.2 Adding the Annotations

After collecting the necessary images (i.e. 5417 images in total). Comes the annotation process, the majority of the dataset does not contain any annotated objects (i.e. bounding boxes of the objects and their classes) and no triplets. In this section we will go through the semi-automatic annotation process.

### 4.2.1 Objects Annotation

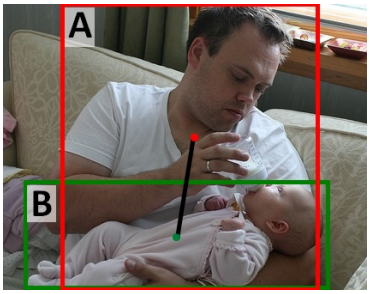
Beside the extracted images from COCO and PASCAL-VOC datasets, i.e. 38.1% of the total images, the remaining portion does not contain any kind of object annotations. To annotate all the 80 categories present in COCO [56]. First, we used Facebook AI Research's Detectron [19], a software providing the state-of-the-art modern object detectors impemented in `Caffe2` framework with a simple Python interface together with a Model Zoo to download pre-trained models, we used a Faster-RCNN [49] object detector with a ResNet [31] backbone to obtain a primary annotations for the objects, they are then manually corrected using a web based annotation tool (i.e. VGG Image Annotator [11]).

### 4.2.2 Interactions Annotation Format

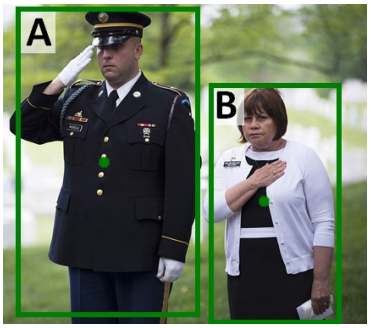
Before adding the H-H interactions annotations, we must first define the format to be followed.

For *<person j, verb, person j>* triplets, person $i$ and person $j$ instances are both represented as bounding boxes, $< x_{\min}, y_{\min}, x_{\max}, y_{\max}, \text{bbox}_{id} >$. $x_{\min}, y_{\min}, x_{\max}$ and $y_{\max}$ represent the rectangle coordinates of the top right corner and the left bottom corner of the bounding box in the image and $\text{bbox}_{id}$ is an unique `ID` generated for every box object in the dataset. To be used in training as box labels.


(a) Commutative.
*<A, highfive, B>*
*<B, highfive, A>*




(b) Non-Commutative.
*<A, holding, B>*
*<A, feeding, B>*




(c) Individual.
*<A, saluting, ∅>*
*<B, saluting, ∅>*

Figure 4.1: An example of the triplets added for each case. **Left**. A commutative interaction, represented with two triplets. **Middle**. An non-commutative case, only one triplet per interaction is need. **Right**. Individual Action are performed, the triplets added do not contain any object instances.

Subsequently, we define the following cases (See Figure 4.1 for some examples):

**- Commutative interactions.** If both the *subject* and *object* are preforming the same action on each other, the interaction taking place is of commutative type, e.g. two persons A and B are having a *handshake* are both participating in and performing the action of *handshake*. Hence, in this instance the interaction is represented as two triplets : *<A, handshake, B>* and *<B, handshake, A>*. A commutative interaction can also take place with multiple actors, for instance, two persons, A and B are *hugging* a third person C, A and B are in a commutative interaction with C, but not in between them, this is represented as $2 \times 2 \times 1 = 4$ triplets: *<A, hugging, C>*, *<B, hugging, C>*, *<C, hugging, A>*, and *<C, hugging, B>*. The maximum number of triplets per interaction is $N_{subjects} \times N_{object} \times 2$.

**- Non-Commutative interactions.** As opposed to a commutative action, if the interaction taking place is only carried out by the *subject*. The maximum number of triplets representing the insteraction is $N_{subjects} \times N_{object}$.

**- Individual interactions.** If the *subject* is performing an individual action, i.e. no *object* instances. The total number of triplets simply equals $N_{subjects}$ and the *object* field in the triplet is left empty (i.e. *<subject, verb, ∅>*).

### 4.2.3 Triplets Annotation

Unlike the objects annotation process, for which there is many available tools, to add the triplets for annotating the interactions we need to do heavy adjustments source code of these tools to meet our needs. Instead, we choose to design our own tool. With two main goals in mind, speed and simplicity, our design requirements were simple.

- Display the images with the objects bounding boxes (from Section 4.2.1) with the ability to display only one category of boxes (e.g. in our case we only work with person boxes).
- Load the verbs and add them to a sidebar list.

- Add mode shortcuts, `s` for Sujects selection mode, `d` for Objects selection mode, `f` for Commutative selection mode, `Enter` for applying the current changes, `ESC` for ignoring them and a `Reset` option to delete all the currently added triplets.
- Display a list of all the triplets added to the current image, with options to delete and inverse the triplet (i.e. subject $\leftrightarrow$ object).
- Add coloring for the bounding boxes of each instance in the image. `White` if the instance is idle (i.e. does not participate in any interaction). `Green` if the instance is an Subject. `Red` is the instance is an object and `Blue` is the intances plays both roles (i.e. in the case of commutativity).

The triplets are then added for each image, by selecting (i.e. double clicking the box) the subject instances (in `s` mode), the objects instances (in `d` mode) or a commutative instance (in `f` mode). Then the changes are applied with `Enter`. With the possibility to delete or modify the current one in case of errors.

The tool was added as a plugin to an existing annotation tool used in LVIC lab (i.e. ATOMiC), which helped us re-use predefined function for displaying the images and the bounding boxes. The coding was done using primarily *TypeScript* (i.e. a static language with a syntaxical superset of *JavaScript*) with some *HTML* and *CSS* coding. In terms of libraries, we used *Polymer* for managing the plugin, using custom elements and running a local server, *PixiJS* for the graphics, *Gulp* for building the project and *Bower* as a package manager. Refer to Appendix C for a design overview.

**Adding the annotations.** By using the labels in the original H-H interactions datasets and the information in Table 4.3 and 4.4, we were able to do a first automatic annotation by choosing the two biggest boxes and creating the triplets (i.e. randomly assiging a Subject and an Object, and if the verb is commutative add them both). The annotations are then corrected and the missing triplets were added using the created annotation tool. Thus reducing the annotation time significantly.

### 4.2.4 Finalizing the Dataset

The final format of the dataset contain, a set of images, objects annotations, interactions annotations and metadata. The annotations and the metadata are provided in three formats (see Appendix B):

- The original format (OSDX) used in the annotation process, OSDX is an internal format specific to the LVIC lab, which is based on Google's Protobuf format, useful for serializing structured data and multi-languages code generation.
- Visual Genome [35] annotation format for its simplicity.
- VCOCO [21] annotation format. This is the main format to be used in the training and evaluation process, taking advantage of the evaluation API already provided by [21].

**Dataset Partitioning.** The dataset was split randomly into two sets, **training** set and **validation** set, with 80% and 20% respectivly. The split was performed randomly while maintaining similar verbs distribution in both sets. For the **test** set, we were able to extract $\sim$ 300 images from Visual Genome [35] dataset using the provided triplets. If the predicate is one of our chosen verbs, and both the subject and

object are of person category (i.e. Wu-Palmer Similarity is calculated for the subject and the word *person*, if it is above a certain threshold the word is considered a synonym of *person*, e.g. guy, girl, etc.), the image is then extracted together with the annotations and converted to the same format of the dataset, the test set in not included in the original dataset due to some annotation errors but can be used as a last stage testing.

## 4.3 Dataset Statistics

In this section we will present some statistics about the H-H interactions dataset. For some annotated examples please refer to Appendix C

**A Comparaison** Our dataset is vastly superior in terms of size, the provided annotations and number of verbs to other Human interaction datasets, and comparable to the state-of-art H-O interactions datasets (Table 4.5).

| Datasets | Type | Size | Verbs | # Objects | Triplets | Boxes of S&O | Object Boxes |
|---|---|---|---|---|---|---|---|
| ShakeFive2 [14] | Videos | 153 | 8 | - | ✗ | ✗ | ✗ |
| Human Interaction Images [13] | Images | 1972 | 9 | - | ✗ | ✗ | ✗ |
| SBU Kinect Interaction [65] | Videos | 171 | 8 | - | ✗ | ✗ | ✗ |
| BIT-Interaction [34] | Videos | 400 | 7 | - | ✗ | ✗ | ✗ |
| UT-Interaction [47] | Videos | 60 | 6 | - | ✗ | ✗ | ✗ |
| TV Human Interaction [43] | Videos | 350 | 5 | - | ✗ | ✗ | ✗ |
| **Ours** | **Images** | **5.5K** | **25** | **78** | ✓ | ✓ | ✓ |
| HICO-DET [63] | Images | 47K images | 117 | 80 | ✓ | ✓ | ✗ |
| VCOCO [21] | Images | 10K images | 27 | 80 | ✓ | ✓ | ✓ |

Table 4.5: Comparaison between our dataset, H-H interactions and H-O inte ractions datasets.

**Dataset Sources** The Dataset contains almost equal proportion form each one of the three sources (H-H interactions and Object detection datasets, and the web). See Figure 4.2

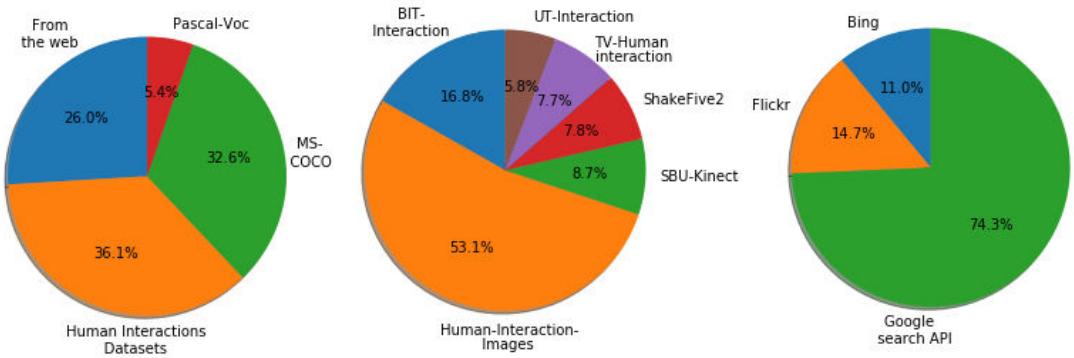



Figure 4.2: The sources of the images in our dataset. **Left**. The dataset contains images from either H-H interaction datasets, Object detection datasets (i.e. MS-COCO and PASCAL-VOC) or the web. **Middle**. The proportions of the images extracted from H-H interaction datasets. **Right** The proportions of images extracted from the web.

**Key numbers** Some basic statistics of the created dataset are:

- Total size: **862 MB**,
- Total number of images: **5417**,
- Average image size (px): **908 x 688**,
- Minimum image size (px): **213 x 142**,
- Maximum size of the image (px): **7500 x 5000**.
- Total number of verbs: **25**,
- Total number of triplet: **10854**,
- Total number of segmentations: **17639**,
- Total number of b-boxes: **37713**,
- Total number of people instances: **21721**.

**Interactions distribution** The total numbers of triplets is 10854, distributed as shown in Figure 4.3. We note that these number are not reflective of the interactions distribution, given that commutative verbs have at least double the count (e.g *Handshake*, *Playing*, *Holding*, etc.).

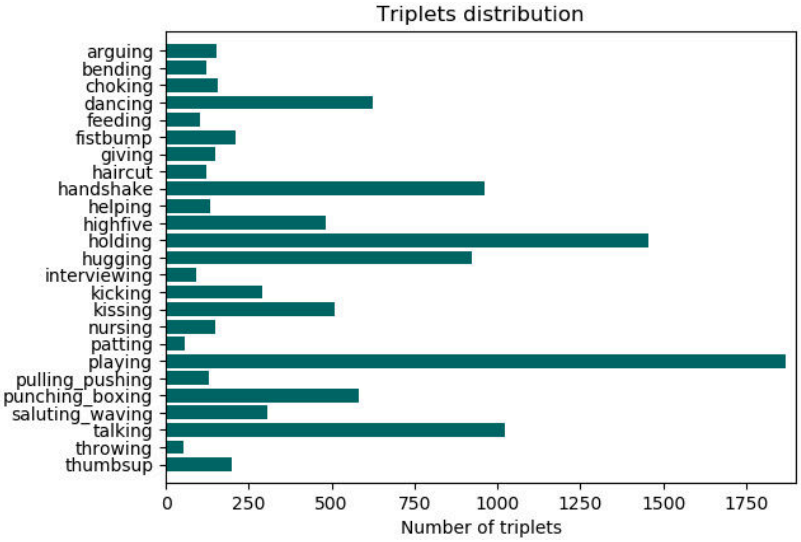



Figure 4.3: The distribution of triplets across all the verbs. Some verbs (i.e. commutative ones) contain more triplets due to the fact that each example is represented by at least two triplets. Thus, we see a lot of commutative verbs dominating the distribution, e.g. *Handshake*, *Playing*, *Holding*. We note that some verbs are in fact rare, mainly due to the difficulty of finding good examples from the sources we used to collect images, e.g. *Throwing* and *Arguing*.

After creating a rich dataset for H-H interactions, and having the training data we need. The next step is to design a model capable of detecting the interaction taking place in still images and taking advantage of the new dataset. The model to be introduced in the next chapter has, as one of its requirements, the capability to detect both H-H and H-O interactions with minor tweaks.

# Chapter 5

# Detecting Human-Interactions

After introducing all the necessary building blocks to design our model and building the dataset for training and evaluation. In this chapter we will introduce our method for detecting Human interactions (i.e. The model is capable of detecting both H-H and H-O interactions). A one-stage model for joint-object detection and interaction recognition. We note that this work is still in progress, and is currently being developped by the members of LVIC's Scene Analysis team and changes on a daily basis. Thus, we explain the main hypothesis behind such design and give an overview of the architectural blocks of the model, the current tasks underdevelopment, necessary details and the possible additions in the near future.

**Disclaimer.** This chapter is the results of joint efforts by LVIC's Scene Analysis team members, Mr.Bertrand Luvison, Ms.Astrid Orcesi and Ms.Sanaa Chafik. In which I participated in.

## 5.1 Overview

Similarly to InteractNet [17] approach, our main objectives are: (1) to design a model capable of managing the $\mathcal{O}(n^2)$ complexity due to all the possible combinations of the detected objects (i.e. any given *subject* can be interacting with any of the detected *objects*), and (2) to use the extracted features from both the objects and the subjects, and their relative spatial positions. But contrary to [17], we would like to design a one-stage interaction model based on an object detector (i.e. RetinaNet [55]), instead of using 3 seperate branches (i.e. first a two-stage object detector [49], and then two branches for action detection and interaction recognition).

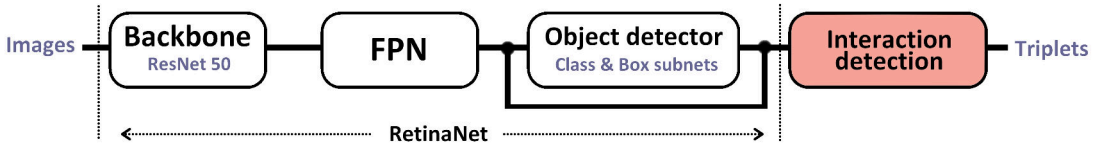



Figure 5.1: A high-level diagram of the proposed method. The model Contains two sub-modules, an object detector, i.e. RetinaNet with its three components, a backbone, a feature pyramid network and the object detection subnets. The detected objects are than fed into the interaction module together with the features computed within the FPN for interaction detection.

These two objectives are satisfied in a similar manner to one-stage detectors (refer to Section 3.1.3), the idea is to generate all the possible interactions directely using the outputs of the object detector in a

fully convolutional way, without employing different intermediate stages for each subtask (i.e. detecting the subjects and then the objects as in [17]).

The object detection module is identical to RetinaNet (please see Section 3.1.4 for details) with a ResNet-50 [31] (i.e. an image classification network trained on ImageNet, 50 refers to the number of layers, other versions such as ResNet-101 and ResNet-151 can be used). The outputs of the object detector (i.e. the anchors classes and box regressions) are concatenated with the FPN features. In the rest of this chapter we will describe the interaction module (see Figure 5.1) in details.

## 5.2 The Model

We take a different approach than [17] (i.e. first predict the *object* location using only the *subject* features and then validate the predicted location) and than [63] (i.e. use the *object* and *subject* feaures and their spatial positions in separate parallel branches). Based on our hypothesis that the detected objects, together with the features and their relative positions provide the necessary information, we suggest to detect the actions in a fully convolutional manner given that the types of objects present in a scene and the relative poses of human instances can give strong cues on the interactions taking place. Hence, we propose a one-stage approach for dense interaction recognition using as inputs stacked feature maps from the FPN with the outputs of RetinaNet, i.e. anchor classes and regressions, hoping that the model will be able to learn the true interactions between all the anchors using a succesion of convolutions.

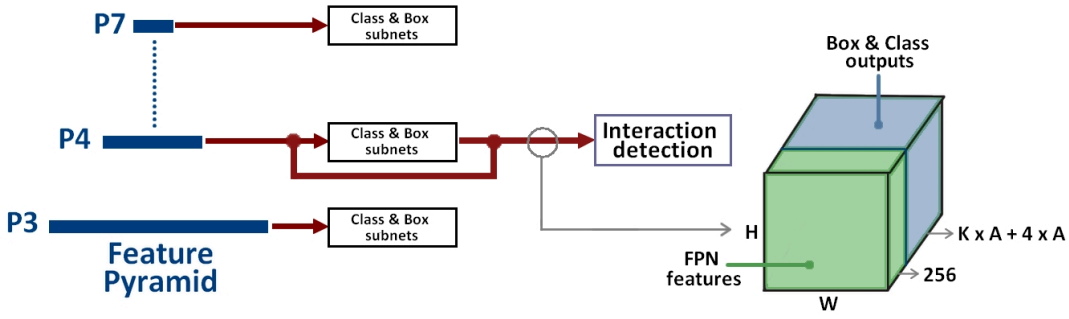



Figure 5.2: The inputs of the interaction module. In this illustration we are only using the level P4 for providing the inputs for the interaction module, the FPN feature maps of size $W \times H \times 256$ are concatenated with the outputs of class & box subnets for object detection of size $W \times H \times (KA + 4A)$, i.e. the spatial dimensions in a given level $W$ and $H$ depend on the input image sizes. The interactions are then detected in a fully convolutional manner.

**The inputs.** For a given feature pyramid level, the dimensions of the feature maps are $W \times H \times 256$, and using the same level for detection, the outputs of the class and box subnets of RetinaNet are of size $W \times H \times (KA + 4A)$, i.e. for each anchor $A$ (i.e. bounding box proposal) a one-hot vector of $K$ classes (e.g. 80 classes +1 for background in our case, such as [56]) indicating the class the anchor is assigned to, along with 4 regression offsets $\triangle(x, y, w, h)$ for correcting the position coordinates from the anchors to the ground-truth boxes. The spatial dimensions $W$ and $H$ depend on the size of the input image and the pyramid level (e.g. with an input image with dimensions $w$ and $h$, for level P3, the feature maps dimensions are $W = \frac{w}{8}$ and $H = \frac{h}{8}$). Given that our goal is to base the detection on all scales, i.e. the interactions taking place in between anchors from different pyramid levels, but due to the complexity of merging all the levels with different spatial dimensions (e.g. for a $512 \times 512$ input image, P3 : $64 \times 64$,

P4 : $32 \times 32$,..., P7 : $4 \times 4$). First, for simplicity we will restrain the detection to only using the pyramid level $P4$ with dimensions $32 \times 32$ for a $512 \times 512$ input image, and to compensate for the lack of various anchors sizes, we use 15 anchors scales (i.e. $2^0, 2^{\frac{1}{3}}, ..., 2^{\frac{13}{3}}, 2^{\frac{14}{3}}$) as opposed to 3 scales (i.e. $2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}$) in the normal case, resulting in a total number of anchors equal to $32^2 \times 15 = 46080$, close to the original count of 49104 anchors, but the main difference is having $15 \times 3 = 45$ anchors per $32 \times 32 = 1024$ positions in the original image, instead of having a smaller number (i.e. 9 anchors) spanning a larger number of positions. But this is only a shortcut to get the first benchmarks of our method. Next, we will suggest three ways to merge the pyramid levels and use the original anchors for detecting the interactions.

**Architecture & Outputs.** The interaction module, similarly to RetinaNet [55], contains two subnets. A Tag subnet and a Score subnet with similar structures. A series of convolutions (i.e. either $1 \times 1$ to adjust the depth or $3 \times 3$) are applied to the input volume, each CONV layer is followed with a batch-normalisation layer [24]. The last CONV layer for each subnet outputs a volume of size $W \times H \times (N_{verbs} \times A)$.

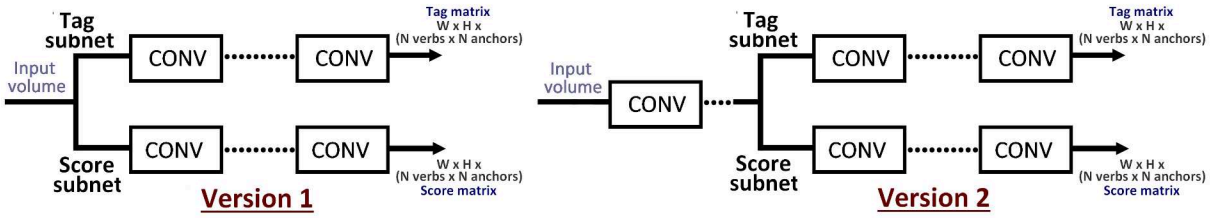



Figure 5.3: The general architecture of the interaction module. Displaying the two versions we are currently considering. Either start directly by two parallel subnetworks with disjoint convolutions (either $1 \times 1$ or $3 \times 3$ convolutions), one for predicting the tags and the other for the scores. Each subnet outputs the corresponding matrix. Or apply a series of convolutions first in a joint manner until the last layers, in which the module branches into two subnets.

**Learning objectives.** Based on two learning objectives we can predict the interactions. (1) Group all the anchors in the same actions together; i.e. minimise the distance between the anchors participating in the same action and maximise the distance between the grouped anchors and the remaining ones. (2) Label (i.e. a one-hot label) the subjects with their respective verbs, i.e. if a given anchor is playing the role of a subject for a given interaction, a vector of length $N_{verbs}$ with all elements equal to 0 except the target verb the subject is assigned to. Knowing the grouped anchors and the subjects of a given the group, we can infer the triplets, and thus detecting the interactions.

The outputs of the interaction module are two matrices. A **Tag** matrix of size $N_{Anchors} \times N_{verbs}$ in which the grouped anchors are assigned the same tag, and a **Score** matrix of the same size $N_{Anchors} \times N_{verbs}$ indicating the anchors playing the role of *subject* for a given verb.

**Example.**

**Triplets**:

<A1, V2, A2>
<A3, V1, A1>
<A5, V4, A4>

**Score Matrix:**

| | V1 | V2 | V3 | V4 |
|---|---|---|---|---|
| A1 | 0 | 1 | 0 | 0 |
| A2 | 0 | 0 | 0 | 0 |
| A3 | 1 | 0 | 0 | 0 |
| A4 | 0 | 0 | 0 | 0 |
| A5 | 0 | 0 | 0 | 1 |

**Tag Matrix:**

| | V1 | V2 | V3 | V4 |
|---|---|---|---|---|
| A1 | $\gamma$ | $\alpha$ | 0 | 0 |
| A2 | 0 | $\alpha$ | 0 | 0 |
| A3 | $\gamma$ | 0 | 0 | 0 |
| A4 | 0 | 0 | 0 | $\beta$ |
| A5 | 0 | 0 | 0 | $\beta$ |

For instance, let's consider the example above with five anchors (A1 ... A5) and four verbs (V1 ... V4) representing three triplets. The tags $\alpha$, $\beta$ and $\gamma$ are real values indicating the anchors that are grouped together. For instance, from the example above, we can deduce that the anchors A1 and A3 with a tag $\gamma$ are both participating in the same action V1. The score matrix is then used to identify the subject, in this case, the score of anchor A3 for verb V1 is 1, thus, A3 is the subject in this interaction and the triplet can be written as <A3, V1, A1>.

## 5.3 Loss Function

In this section we will present the loss function our model is currentely based on. The final version of the loss function is still underconsideration and the one with the best perfomances is to be chosen. The total loss function is :

$$L_{total} = L_{interaction} + L_{obj\_detection} \tag{5.1}$$

$$L_{total} = (L_{box} + L_{tag}) + (L_{class} + L_{regression}) \tag{5.2}$$

$L_{class}$ is the focal loss used in [55], $L_{regression}$ is a Huber loss between the regression targets and the predictions. Now we will discuss $L_{tag}$ and $L_{box}$ used in the interaction module.

**Tag loss.** As mentioned above, the learning objective is to group the anchors participating in the same interaction together. This is achieved by imposing a grouping loss based on the associative embedding introduced in [41] and [2]. The grouping loss assesses how well the predicted tags agree with the ground truth grouping. Specifically, we retrieve the predicted tags for all interactions of all the anchors; we then compare the tags within each interaction and across them. Tags within an interaction should be the same, while tags across should be different.

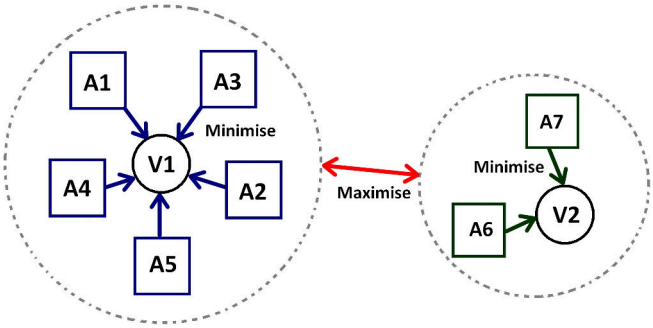



Figure 5.4: The objective of tag loss is to penalize the distance between the anchors in the same interaction and penalize the complement of the distance between anchors performing different actions. Such that to minimise the intra-distance and maximise the inter-distance of the grouped anchors. In this example we would like to minimise the tag distance between the anchors performing actions V1 and V2 and maximise their inter-distance.

The tag loss contains $L_{pull}$ to minimise the tag intra-distance between the grouped anchors and a $L_{push}$ to maximise the inter-distance.

$$L_{tag} = L_{pull} + L_{push} \tag{5.3}$$

Rather than enforce the loss across all possible pairs of anchors, [41] use a reference tag for each interaction which is simply the mean of the tags of the anchors to be grouped together.

$$\bar{h}_n = \frac{1}{K} \sum_k h_k(x_{nk}) \tag{5.4}$$

Using Equation 5.4, and within each individual interaction, we compute $L_{pull}$ as the squared distance between the reference tag $\bar{h}_n$ and the predicted, tag for each anchors $k$, and for each interaction $n$.

$$L_{pull} = \frac{1}{N} \sum_n \sum_k (\bar{h}_n - h_k(x_{nk}))^2 \tag{5.5}$$

And then, $L_{push}$ is computed between pairs of interaction $n$ and $n'$, we compare their reference tag to each other (i.e. $\bar{h}_n$ and $\bar{h}'_n$) with a either: (1) a penalty that drops exponentially to zero as the distance between the two tags increases [41] or (2) a margin-based penalty as in [2] with a margin $m$ (e.g. $m = 8$). We are currentely testing both versions.

$$L_{push\ exp} = \frac{1}{N^2} \sum_n \sum_{n'} \exp\left\{ \frac{1}{2\sigma^2} (\bar{h}_n - \bar{h}'_n) \right\}^2 \tag{5.6}$$

$$L_{push\ margin} = \sum_n \sum_{n'} \max\left(0, \|\bar{h}_n - \bar{h}'_n\|\right) \tag{5.7}$$

Defining a loss function based on both $L_{push}$ and $L_{pull}$. We hope that optimizing it, the anchors participating in the same action are clustered together as a result.

**Score loss.** The score comes into play when we want to differentiate between the anchors within the same group (i.e. having similar tags). Anchors with a score equal to 1 for a given verb are interpreted as the subjects for given interaction. The loss in this case is straightforward using either an $L2$ loss or a binary cross entropy loss between the predicted scores $\hat{S}_k$ and the ground-truths $S_k$ for all positive anchors $k$ (i.e. anchors that are not classified as background).

$$L_{score\ L2} = \sum_k (S_k - \hat{S}_k)^2 \tag{5.8}$$

$$L_{score\ CS} = \sum_k -(1 - S_k) \log\left(1 - \hat{S}_k\right) - S_k \log\left(\hat{S}_k\right) \tag{5.9}$$

**Optimization.** We use Momentum update (refer to Section 2.3 for more details) to update the weights and optimize the total loss (see Equation 5.1). For further implementation details, see Appendix B.

## 5.4 Merging the FPN

Using RetinaNet as the object detector of choice in this work imposes some difficulties, mainly the different spatial dimensions of the outputs. In the first place we used only one level for providing the inputs to interaction detection module (e.g. P4 with spatial dimensions of $32 \times 32$ for an $512 \times 512$ image). But to detect the interactions taking place between scene objects of different sizes the need to merge to feature pyramid of the FPN rises. We identified three possible solutions.

**Autoencoder** Autoencoders are an unsupervised learning technique in which we leverage neural networks to impose a bottleneck in the network which forces a compressed knowledge representation of the original input. In our case, we'd like to compress the spatial dimensions of the bottom level P3 down to $32 \times 32$ and enlarge it for the top levels, and then stack them in one volume containing both the feature maps and the class & box subnet outputs. The model is trained to compress all the inputs down to $4 \times 4$ and decode them up to $64 \times 64$. Depending on the levels, the layer we place the features in the autoencoder varies (see Figure 5.5), but the output is the same and of size $32 \times 32 \times 25$. Thus, the final size of the input to the interaction module is equal to $32 \times 32 \times (5(\text{levels}) \times (KA + 4A + 256))$. The diffiulties this approchad may impose is the complexity of training and deploying the autoencoder in such a setting.

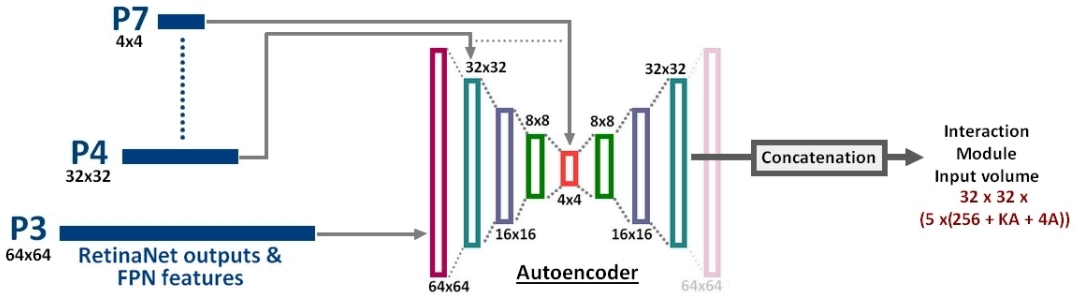



Figure 5.5: Depending on each level, the feature maps are forwarded to a different input layer of the autoencoder. The objective is to obtain an output of the same spatial dimensions of size $32 \times 32$. Using the resulting volume as an input, giving the interaction module the ability to detect the interaction between anchors of different sizes.

**Upsampling.** A simpler top-down approach, is applying a series of nearest neighbor upsampling, starting from the smallest level P7 down to P3. In the start we upsample the feature maps and RetinaNet anchors of P7 with a factor of 2 to have the same spatial dimensions of P6, after that we apply a number of consecutive convolutions. The results are then also upsampled to get to the sizes of P5 and so on. In the final step we will obtain a volume of the same sptial dimensions of P7 and contains rich information extracted from all the levels due to the convolutions applied. This is the input to the interaction detection stage. This approach appears to be simple, yet effective.

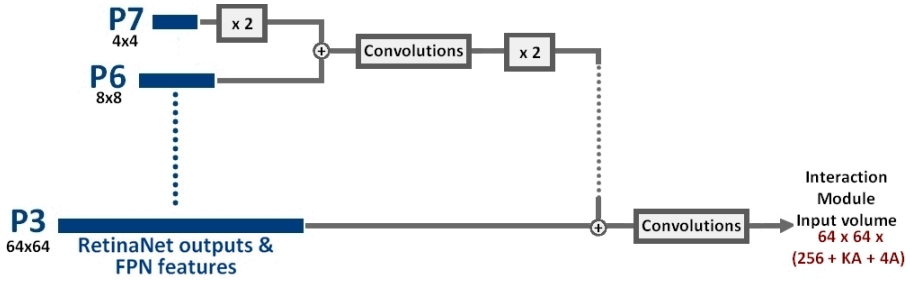



Figure 5.6: Starting from the top level of spatial dimensions $4 \times 4$, a nearest neighbor upsampling is applied to it, ending up with the same dimensions as the bottom level, we then concatenate both volumes and apply a series of convolutions to extract the meaningful information. The same process is then applied in a successive manner down to the last level. The resulting volume is our input to the interaction module.

**Concatenation.** As opposed to upsampling, we can use a similar bottom-up approach in a series of concatenations. Starting from the lower pyramid level P3 we concatenate it with the P4 resulting with a

volume of the same spatial dimensions as P4 and 5 times the depth. A series of convolutions are applied to reduce the depth to $(256 + KA + A)$ from $5 \times (256 + KA + A)$, and we go through same process again until the last level P7. Similar to the upsampling method, this approach is characterized by its simplicity, but with one apparent drawback, the small size of the resulting volume, giving the interactions modules limited features to the detection.

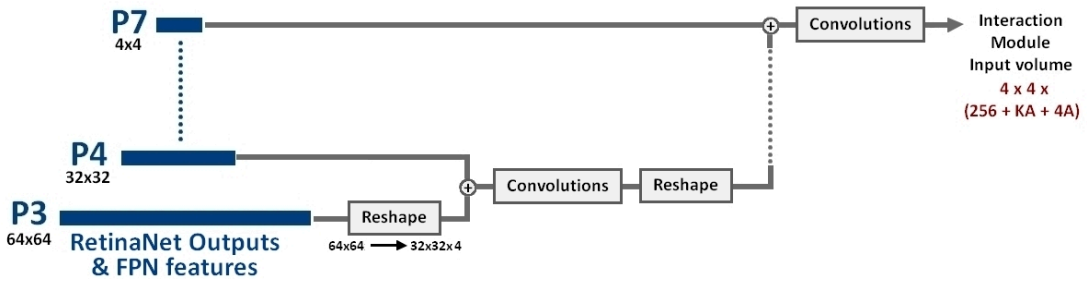



Figure 5.7: A bottom-up approach, starting from the biggest level P3 (i.e. containing both the FPN features and the outputs of RetinaNet). After it is reshaped into a volume with the same spatial dimensions as the top level and 4 times the depth. The result is concatenated with the upper levels and a series of convolutions are applied to the outcome. After a repetition of these operations up to the last level, we end up with a small volume of size $4 \times 4 \times (256 + KA + A)$. This is the input to the next stage.

## 5.5 Human-Human *vs.* Human-Object interactions

During the development process, the model was being developed simultaneously for both, H-H and H-O interactions detection, using VCOCO [21] dataset. The differences between detecting H-H and H-O interactions are very subtle. The main one comes into play in the inference stage when we chose the types of objects to take into consideration. For H-H interactions we only take into consideration human instances as possible *object* condidates, in the other hand, for H-O interactions, only object instances are taken as prospective candidates. One additional distinction between VCOCO and H-H interactions datasets, is the fact that in H-H interactions dataset a same person instance can have multiple interactions of the same type (i.e. the same verb in the same scene. e.g. A *hugging* B and A *hugging* C), which is not the case in VCOCO dataset.

A supplementary remark is that for H-O interactions, in addition to the possible triplets : <*subject*, verb, $\varnothing$> and <*subject*, verb, *object*>. A third case rises : <*subject*, verb, *object*, *instruments*>.

All these subtle differences must be taken into consideration when using the proposed method for either detecting H-H or H-O interactions, and the necessary adjustments must be made beforehand.

## 5.6 Experiments

In this section we present some results for different version of our model and using different losses and hyperparameters.

**Datasets.** For training and evaluating our model, we mainly used the dataset set we introduces in Chapter 4. With $\sim$ 4500 images for training and $\sim$ 1000 images for evaluation. The model was also trained and eveluated using VCOCO dataset.

**Data Preprocessing.** All the input images are re-sized into $512 \times 512$ images. A data augmentation technique can be used, such as only taking one interaction per time by taking the union of the spatial extent of the subject and object boxes and taking a corresponding crop form the image.

**Hyperparameters.** We test different number of convolution in both the Tag and Score subnets, varying from 7 to 4 CONV layer with either $3 \times 3$ or $1 \times 1$ convolutions. With a batch of size 32 images and an initial learning rate of 0.04.

The results obtained for various experiments are as follows (the metrics used are described in Section 3.2.4).

| | Model | | | | Dataset | Interactions | AP agent | AP role |
|---|---|---|---|---|---|---|---|---|
| | FPN | Interaction Subnet | $L_{tag}$ | $L_{score}$ | | | | |
| | | Model C of [21] | | | VCOCO | H-O | 65.1 | 37.8 |
| | | InteractNet of [17] | | | VCOCO | H-O | **69.2** | **40.0** |
| **Ours** | One level | Joint $1 \times 1$ | Margin | CS | VCOCO | H-O | 21.64 | - |
| **Ours** | One level | Joint $1 \times 1$ | Margin | CS | VCOCO | H-O | 21.07 | - |
| **Ours** | One level | Joint $1 \times 1$ | Exp | CS | VCOCO | H-O | 22.45 | - |
| **Ours** | One level | Joint $1 \times 1$ | Exp | CS | VCOCO | H-O | 29.92 | - |
| **Ours** | One level | Joint $1 \times 1$ | Margin | L2 | Ours | H-H | 8.5 | - |

Table 5.1: The results of the interaction detection for different loss function, hyperparameters and model architecture. We note that this is still a work in progress and these are not the final results.

## 5.7 Future Work

One natural way to extend the problem is to detect all the interactions of both types with some minor changes. This time around we need to test for all possible instances as candidates for *objects* in an interaction.

Given the differences between the two tasks, with all the possible triplets forms in H-O interactions together with the commutative actions of H-H interactions, and the other distinctions mentioned in Section 5.5, the complexity of the problem increases dramatically. But it still remains an interesting one to tackle.

**All interactions, a new benchmark** Due to the fact that both, the interactions dataset and VCOCO have the same structure. We then introduce a new benchmark for detecting all the interactions.

| Datasets | Size | Verbs | # Objects | Triplets | Boxes of S&O | Object Boxes |
|---|---|---|---|---|---|---|
| Ours | 5.5K | 25 | 78 | ✓ | ✓ | ✓ |
| VCOCO [21] | 10K images | 27 | 80 | ✓ | ✓ | ✓ |
| **All interactions** | **15.5K** | **52** | **80** | ✓ | ✓ | ✓ |

Table 5.2: A dataset to be use for detecting all interactions, both H-H and H-O interactions.

This dataset can then be used for training a model, with a similar architecture to the proposed one, to detect all types of interactions taking place in still images.

# Chapter 6

# Conclusions

Progress on core computer vision tasks (e.g image classification) has energized the field and unlocked a wide variety of ambitious problems that once defied our attempts but suddenly seem within our grasp. In particular, in this work we developed a model that pushes the frontier of visual recognition by expanding the label space from a discrete set of single categories to the space of interaction detection in the from of triplets. We argued that doing so is 1) a stepping stone towards Artificial Intelligence agents that can perceive the visual world in richer manner and have a deeper semantic understanding, 2) a yet unsolved and critical next frontier in Computer Vision and 3) desirable from a practical perspective and can be used for various application such as image search and retrieval, caption generation and visual question answering.

Concretely, after an overview of the field and a state-of-the-art presentation. In Chapter 4 we introduced a new dataset for human interactions, this datasets is by far, to the best of our knowledge, the largest and the most diverse one available for Human-Human type of interactions. The dataset together with the developed model are to be a published in the near future.

In Chapter 5 we developed a model for detecting the interactions taking place in the form of triplets *<subject, verb, objet>* in a given image. A model characterized with its one-stage approach for joint object and interaction detection. Giving great potential for faster inference time, simple end-to-end training, and constant computational complexity in comparaison to the state of the art.

In our approached we focused on detecting mainly Human-Human interactions, but our model can also be used for detecting Human-Object interactions with minor ajustments and some results were provided for this task. Thus, for the short-term perspectives of this work, we hope to further tune our model and obtain greater performances and accuracy levels rivaling the state-of-the-art methods in both Human-Human and Human-Object interactions. And to tackle the problem of detecting all types of interaction using a combination of our dataset and Human-Object type datasets.

With this work, we hope that the introduced dataset and many of the architectural elements featured in this work can be reused and help inspire future work.

Despite the recent breakthroughs in visual recognition, it is clear that many challenges still remain before we can realize Turing's vision of machines that have a deep understanding of the visual world and interact with us. The remaining challenges are best illustrated with a concrete example. Let's consider the image in Figure 6.1.

Figure 6.1: An amusing image worth a thousand words.

It only takes a few short moments to go from glancing at the image to grasping a wide range of ideas and obtain a realization of the full depicted situation. The steps we take to fully perceive this visual scene are numerous and astonishing. From recognizing the mere presence of several people in the image, to remarking the setting and the presence of cameras and deducing a possible meeting, to connecting our knowledge of the identities of the actors in the image, the current political situation, their poses and the apparent disappointment towards each other, leading us to further enhance our understanding and giving the image a whole new meaning.

To endow computers with such an understanding of so many interconnected abstract concepts and knowledge, great strides are to be made. And we hope that with our work we were able to push the boundaries, even if it is for an infinitely small amount.

# Bibliography


[1] A. Kembhavi A. Gupta and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *TPAMI*, 2009.

[2] Zhiao Huang Alejandro Newell and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *NIPS*, 2017.

[3] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012.

[4] Yuqi Zhang Bo Dai and Dahua Lin. Detecting visual relationships with deep relational networks. *CVPR*, 2017.

[5] M. Bernstein C. Lu, R. Krishna and L. Fei-Fei. Visual relationship detection with language priors. *ECCV*, 2016.

[6] A. Toshev C. Szegedy and D. Erhan. Deep neural networks for object detection. *NIPS*, 2013.

[7] D. Erhan C. Szegedy, S. Reed and D. Anguelov. Scalable. Scalable, high-quality object detection. *arXiv*, 2014.

[8] John Canny. A computational approach to edge detection. *TPAMI*, 1989.

[9] Yangqing Jia Pierre Sermanet Scott Reed-Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Andrew Rabinovich Christian Szegedy, Wei Liu. Going deeper with convolutions. *CVPR*, 2014.

[10] A. Toshev D. Erhan, C. Szegedy and D. Anguelov. Scalable object detection using deep neural networks. *CVPR*, 2014.

[11] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). `http://www.robots.ox.ac.uk/~vgg/software/via/`, 2016. Version: 1.0.5, Accessed: 15/05/2018.

[12] Eslami S. M. A. Van Gool L. Williams-C. Winn J. Everingham, M. and A Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015.

[13] Eslami S. M. A. Van Gool L. Williams-C. Winn J. Everingham, M. and A Zisserman. Facial descriptors for human interaction recognition in still images. *arXiv*, 2016.

[14] Eslami S. M. A. Van Gool L. Williams-C. Winn J. Everingham, M. and A Zisserman. Spatio-temporal detection of fine-grained dyadic human interactions. *HBU*, 2016.

[15] Yingying Chen Fangchen Liu Mike Liao-Vashisht Madhavan Fisher Yu, Wenqi Xian and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv*, 2018.

[16] Simon Osindero Geoffrey E. Hinton and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *NIPS*, 2015.

[17] Piotr Dollàr Georgia Gkioxari, Ross Girshick and Kaiming He. Detecting and recognizing human-object interactions. *CVPR*, 2018.

[18] R. Girshick. Fast r-cnn. *ICCV*, 2015.

[19] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[21] Saurabh Gupta and Jitendra Malik. Visual semantic role labeling. *arXiv*, 2015.

[22] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, 2017.

[23] T. S. Huang. Computer vision: Evolution and promise. *CERN School of Computing*, 2015.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015.

[25] Theo Gevers-Arnold WM Smeulders Jasper RR Uijlings, Koen EA Van De Sande. Selective search for object recognition. *IJCV*, 2013.

[26] Elad Hazan John Duchi and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 2011.

[27] Chen Sun Menglong Zhu Anoop Korattikara-Alireza Fathi Ian Fischer Zbigniew Wojna Yang Song Sergio Guadarrama Kevin Murphy Jonathan Huang, Vivek Rathod. Speed/accuracy trade-offs for modern convolutional object detectors. *CVPR*, 2017.

[28] Jia Deng Jonathan Krause, Michael Stark and Li Fei-Fei. 3d object representations for fine-grained categorization. *ICCV*, 2013.

[29] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[30] Laurens van der Maaten Li Fei-Fei C. Lawrence Zitnick Ross Girshick Justin Johnson, Bharath Hariharan. Clevr: A diagnostic dataset forcompositional language and elementary visual reasoning. *CVPR*, 2017.

[31] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *CVPR*, 2016.

[32] Andrej Karpathy. Connecting images and natural language. *Stanford's Dissertation*, 2017.

[33] Diederik Kingma and Jimmy Ba. Adam. A method for stochastic optimization. 2015.

[34] Yu Kong, Yunde Jia, and Yun Fu. Learning human interaction by interactive phrases. 2012.

[35] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Arxiv*, 2016.

[36] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. *arXiv*, 2017.

[37] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[38] L. Zettlemoyer M. Yatskar and A. Farhadi. Visual semantic role labeling for image understanding. *CVPR*, 2016.

[39] Luke Zettlemoyer Mark Yatskar and Ali Farhadi. Situation recognition: Visual semantic role labeling for image understanding. *CVPR*, 2016.

[40] Jianmin Chen Zhifeng Chen Andy Davis-Jeffrey Dean Matthieu Devin Sanjay Ghemawat Geoffrey Irving Michael Isard Manjunath Kudlur Josh Levenberg Rajat Monga Sherry Moore Derek G. Murray Benoit Steiner Paul Tucker Vijay Vasudevan Pete Warden Martin Wicke Yuan Yu Martín Abadi, Paul Barham and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. *USENIX Symposium*, 2016.

[41] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. *NIPS*, 2017.

[42] Hao Su Jonathan Krause Sanjeev Satheesh-Sean Ma Zhiheng Huang Andrej Karpathy Aditya Khosla Michael Bernstein Alexander C. Berg Olga Russakovsky, Jia Deng and Li FeiFei. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[43] Marszalek M. Reid I. Patron-Perez, A. and A Zisserman. Structured learning of human interactions in tv shows. *TPAMI*, 2016.

[44] David McAllester Pedro F. Felzenszwalb, Ross B. Girshick and Deva Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.

[45] T. Darrell R. Girshick, J. Donahue and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *ICCV*, 2014.

[46] Michael Bernstein Ranjay Krishna, Ines Chami and Li Fei-Fei. Referring relationships. *CVPR*, 2018.

[47] M. S. Ryoo and J. K. Aggarwal. UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA). `http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html`, 2010.

[48] M. Sadeghi and A. Farhadi. Recognition using visual phrases. *CVPR*, 2011.

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

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[51] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[52] Xinyan Zhu Tao Hu and Wei Guo. Two-person interaction recognition based on key poses. *JCIS*, 2016.

[53] T Tieleman and Geoffrey E Hinton. Divide the gradient by a running average of its recent magnitude. *Lecture 6.5-RMS-Prop*, 2012.

[54] Ross B Girshick Kaiming He-Bharath Hariharan Serge J Belongie Tsung-Yi Lin, Piotr Dollár. Feature pyramid networks for object detection. *CVPR*, 2017.

[55] Ross Girshick Kaiming He Piotr Dollár Tsung-Yi Lin, Priya Goyal. Focal loss for dense object detection. *arXiv*, 2017.

[56] Serge Belongie James Hays Pietro Perona Deva Ramanan Piotr Dollár C Lawrence Zitnick Tsung-Yi Lin, Michael Maire. Microsoft coco: Common objects in context. *ECCV*, 2014.

[57] Paul Viola and Michael Jones. Rapid object dete ction using a boosted cascade of simple features. *CVPR*, 2001.

[58] Na vneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[59] D. Erhan C. Szegedy W. Liu, D. Anguelov and S. Reed. Ssd: Single shot multibox detector. *ECCV*, 2016.

[60] Yoshua Bengio Yann LeCun, Léon Bottou. Gradient-based learning applied to document recognition. *CVPR*, 1998.

[61] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. *CVPR*, 2010.

[62] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv*, 2015.

[63] Xieyang Liu Huayi Zeng Yu-Wei Chao, Yunfan Liu and Jia Deng. Learning to detect human-object interactions. *WACV*, 2018.

[64] Yugeng He Jiaxuan Wang Yu-Wei Chao, Zhan Wang and Jia Deng. Hico: A benchmark for recognizing human-object interactions in images. *ICCV*, 2015.

[65] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L. Berg, and Dimitris Samaras. Two-person interaction detection using body-pose features and multiple instance learning. *CVPRW*, 2012.

[66] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014.

# Appendix A

# Implementation

In terms of the model implementation, the Deep Learning framework chosen in this work is TensorFlow (see Chapter 2). In LVIC lab TensorFlow is the main framework used, with many available libraries (e.g. predefined image classification networks, pre-implemented models such as Faster-RCNN and RetinaNet, pre-tuned optimizers, etc.) and a known efficient workflow. In this work we adhered to the same standards and followed the same implementation pipelines.

**Pre-processing.** First the dataset is converted to `tf_record` files from the row formats (i.e. jpg and JSON files). `tf_records` are a convenient way to handle the complex dataset structure in a single record file. All the data is converted in a squential manner (e.g. filename, image, b-boxes, triplets, etc.) into raw data bytes for faster reading speeds during training. In the data-preprocessing stage all the images are converted to $512 \times 512$ sizes with the necessary padding. A possible data augmentation stage can be applied to take into consideration only one interaction at a time.

**TensorFlow API.** In this work we used the high-level API `tf_estimator`, characterized by its straight-forwardness, in which the process of building the model is abstracted to only specifying the high level structure in `model_fn`, the hyper-parameters (e.g. using a JSON files for simplicity and reproducibility in our case), the optimizer and the functions used for evaluation and data-preprocessing, etc. Without any need to manage the low-level distribution of computations across the availble CPU cores and GPU devices.

**Training.** The training of the model is done using multiple TITAN X (pascal architecture) GPUs with batches of 32 images. The multi-gpu training is handled automaticaly by the `tf_estimator` API, after choosing the GPUs to train on, the model is then copied into all the GPUs, then after a forward pass, the gradients are calculated in each GPU seperately and are then sent to the CPU for aggregation and for updating the weights (i.e. *data parallelism*). Resulting in faster training times.

**Evaluation.** The evaluation is done using VCOCO API (**Link**), The evaluation functions are modified for our particular needs in the Human-Human interactions. The evaluation is run periodically each time a new checkpoint (i.e. a sort of screen-shots of the current state of the model in a given time, containing all the defined tensor shapes, learned weights and the model structure for later use).

# Appendix B

# Annotation Formats

This appendix is intended as a short overview of the annotation formats, generally the annotation formats come as `JSON` files along with the images of the dataset due to their lightweight nature and their simplicity for data interchange, but other formats can also be used such as `XML`, `CSV`, `Protobuf` format or even a simple `txt` file.

**Image classification.** For image classification, a label referencing to a given class (e.g. 0 for dog, 1 for cat, etc.) is assigned to all the images. In the annotation files we may find a dictionary in which the key is the unique ID of the image and the value is the class label. With a possibility of having multiple values such as the main class, category and the subclass labels (e.g. category : animal, class : dog, subclass : german shepherd).

**Objects.** the objects in the image are annotated using bounding boxes, the box coordinates can be specified as $\{x_{min}, y_{min}, x_{max}, y_{max}\}$ or $\{x_{min}, y_{min}, h, w\}$. Along with the class label of the object inside the box. In the annotation files, this can be represented as a dictionary where the keys are the image IDs, and the values are lists, in which each element is also a `JSON` dicionary containing the box coordinates, the class label, a unique ID and other additional elements.

**Keypoints.** Human poses are specified with a number of predefined points or joints (see Figure B.1). Per example for COCO dataset [56], they define 17 points with 19 vertices for connecting them. This is then annotated with a list of the 19 vertices with 2 points in each element. Each person instance with an annotated human-pose have the same ID as the box ID that references the same instance.

**Segmentation.** Represented as polygons, annotated as a single list of successive coordinates (i.e. $X_1$, $Y_1$, $X_2$, $Y_2$ ... $X_1$, $Y_1$). The starting point does not matter as long as the final point is the same. Thus specifying a closed loop as a polygon. Similar to keypoints. The annotated instances share the same IDs with the objects and keypoints of the same instance.

**Interaction.** The interaction are represented as mentioned in Chapter 4 in the form triplets, in which the dictionary values reference the image IDs, and the values are the triplets of the interactions taking place in the image. The *object* and *subject* can be simple IDs referencing to the object ID in the objects annotation (i.e. VCOCO [21]) or may contain the ID, the class labels and the box coordinates.

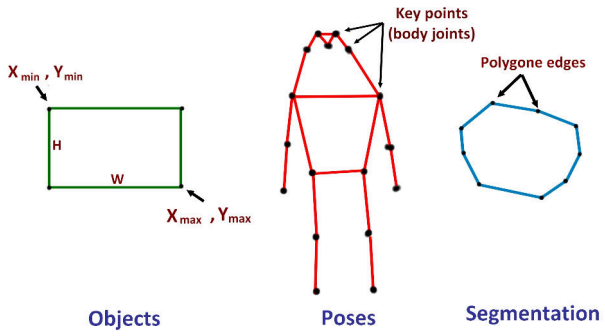



Figure B.1: How the data used in some computer vision tasks is represented.

For the specifics please visit (**Link**) for Visual Genome dataset or (**Link**) for COCO dataset.

# Appendix C

# Dataset. *Design & Examples*

In this appendix we will present the design layout of the annotation tool, together with randomly chosen examples from the created dataset in Chapter 4.

Figure C.1 displays the design of the interaction annotation tool we designed for adding the triplets. The right panel is the main plugin to add the triplets. Showing the current selecting mode, download button for recovering the annotations, adding a verb to choose from, a scroling list to choose the right verb for a given image and options for possible corrections.

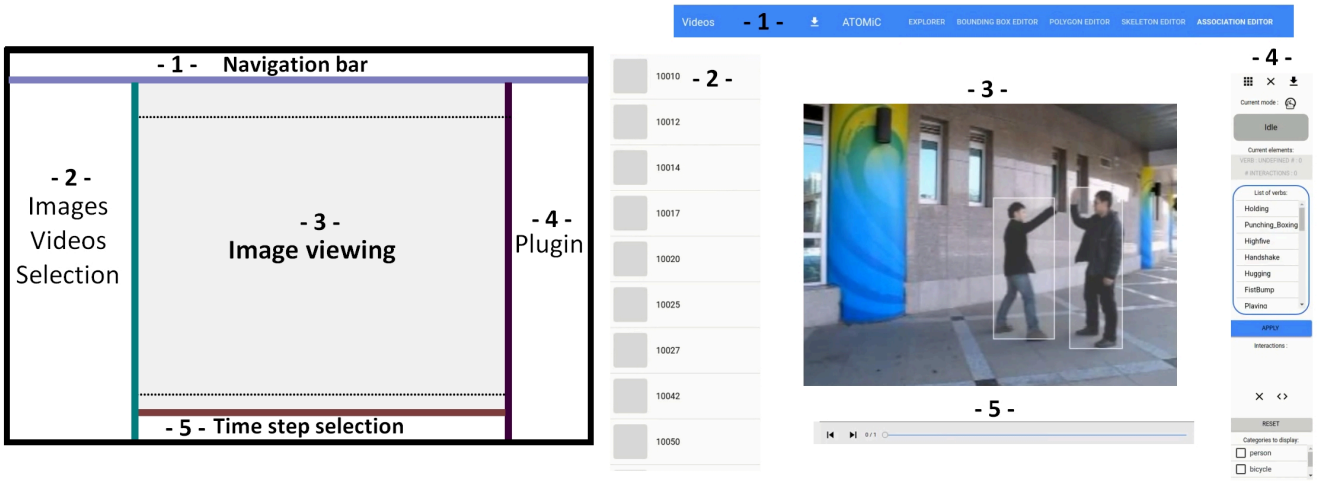
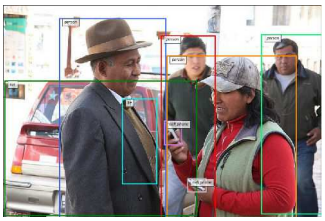
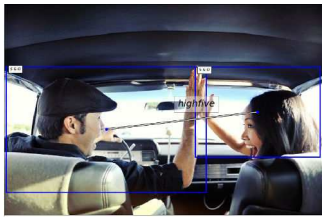
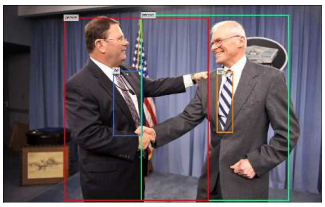


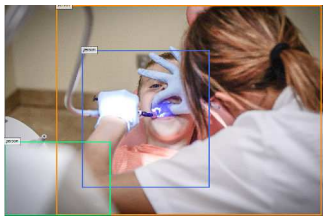
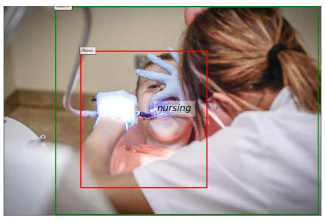
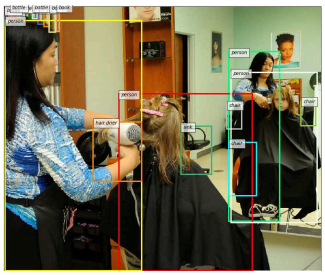

Figure C.1: Design of the interaction annotation tool.

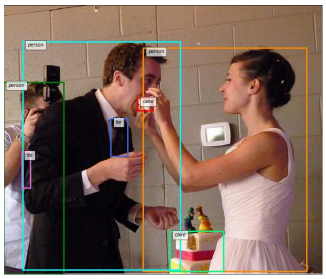
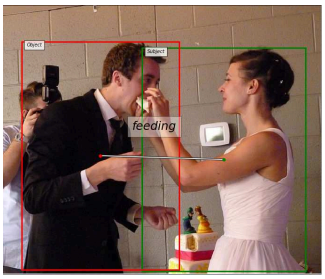
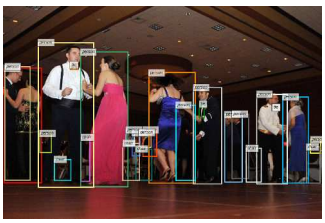
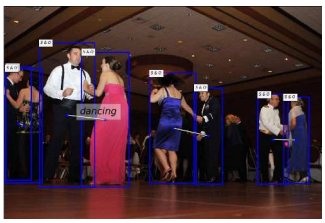
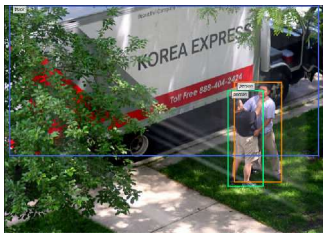
Bellow are some examples extracted from the created Human-Human interactions dataset. Each row contains two pairs of images, in each pair, the right image contains the annotated objects and the left one the annotated interactions.

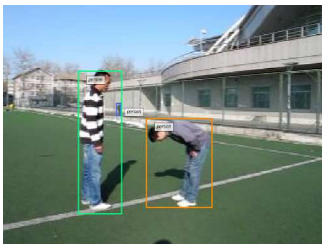


interviewing


highfive


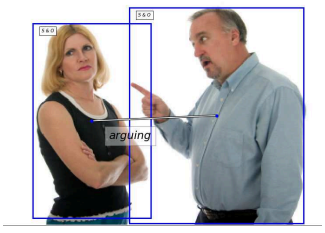
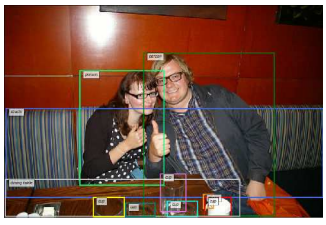


nursing


haircut


feeding


dancing

KOREA EXPRESS

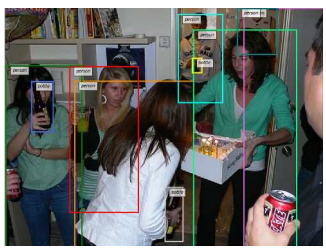
KOREA EXPRESS
choking


bending

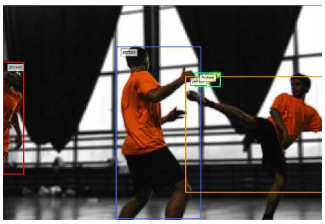
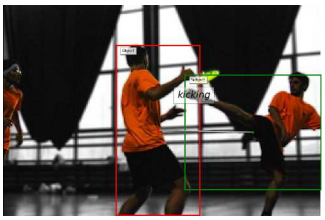
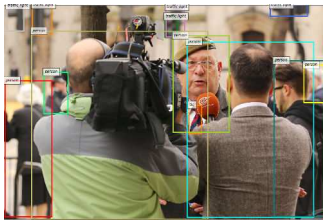
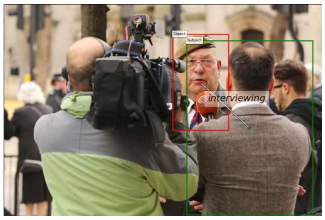
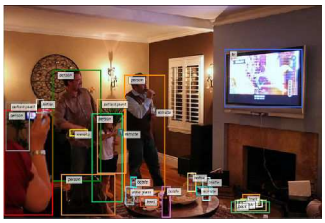
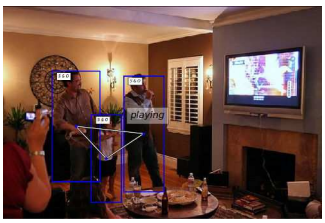
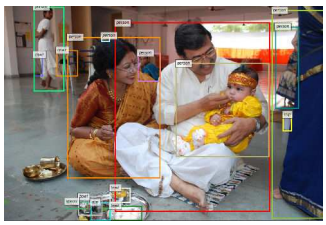

holding


arguing


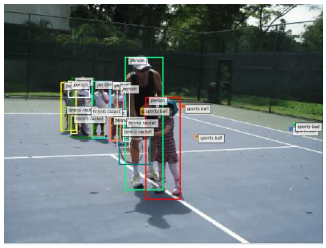
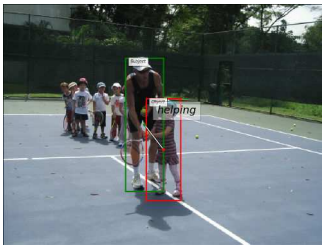
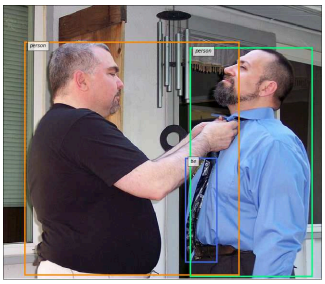
thumbsup

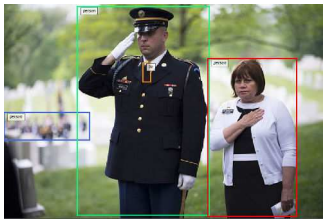

holding
kissing


giving

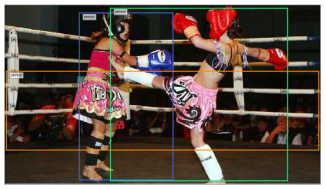


kicking


interviewing


playing


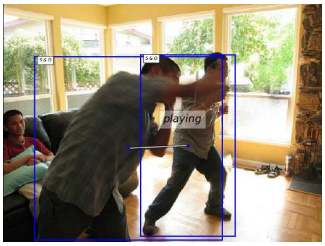
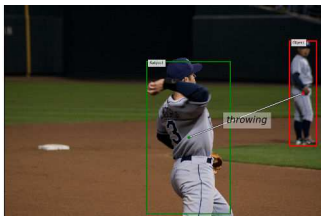
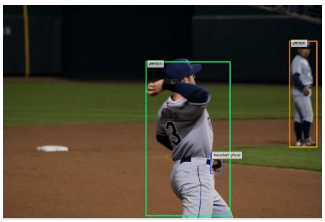
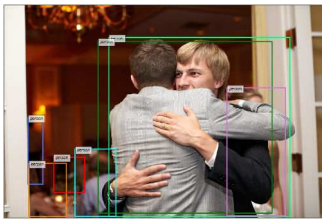
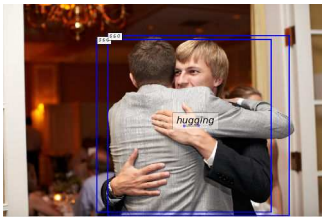
feeding


choking


helping


helping


saluting_waving


giving




playing

throwing



hugging


highfive

**Titre** : Reconnaissance et détection des interactions humaines

**Mots clés :** Vision par ordinateur, Apprentissage statistique, Apprentissage profond, Intelligence Artificielle.

**Résumé** : L'intelligence artificielle a comme but principal de créer des machines dotées d'une intelligence humaine, donnant ainsi aux ordinateurs des capacités cognitives similaires à celles des humains, et pourquoi pas les dépassants un jour. La vision par ordinateur, en tant que branche de l'intelligence artificielle, a pour objectif d'imiter le système visuel chez les humains, capable de transformer un ensemble de particules de lumière arrivant au cortex visuel à une compréhension riche et profonde des scènes. Pour développer de tels agents, capables de transformer un ensemble d'intensités en forme de pixels en une information de haut niveau et riche sémantiquement, des efforts considérables ont été déployés dans le domaine de la vision par ordinateur au cours des dernières décennies, mais seuls des progrès limités ont été réalisés et n'ont pas répondu aux attentes. Et ceci jusqu'à 2012, avec le retour en force d'apprentissage profond et les avancements simultanés en puissance de calculs et la quantité des données disponibles. Les ordinateurs sont désormais capables de réaliser des tâches qui ont été limitées aux humains, comme la classification des images en différentes sous-catégories, la détection des objets présents dans une image, ou la segmentation pixélisée. Mais pour obtenir une compréhension visuelle profonde du monde qui nous entoure à travers, de grands améliorations doivent être faits.

Dans ce travail, nous abordons le problème de la détection et de la reconnaissance des interactions humaines. Nos deux principales contributions sont un jeux de données pour l'interaction humaine et un modèle capable de détecter les objets et de reconnaître les interactions d'une façon conjointe.

**Title:** Detecting and Recognizing Human Interactions

**Keywords:** Computer vision, Machine Learning, Deep Learning, Artificial Intelligence.

**Abstract:** The ultimate goal of Artificial Intelligence is to create a machine with human-like intelligence, thus giving silicon based computers cognitive abilities similar to what humans have, and maybe one day surpassing them in a form of a utopian machine, an AGI (Artificial general intelligence). The long-standing goal of Computer Vision as branch of Artificial Intelligence is imitating the visual system of our brains, going from a row input in the form of photons, transformed in the visual cortex to a rich and deep understanding of the scenes in front of us. To develop such agents, capable of turning pixel intensities into rich semantic information, immense efforts have been put into the field of Computer Vision in the last decades, years ago the field seemed to come to a halt regardless of the amount efforts and research put in, but since 2012 with the resurgence of Deep Learning and simultaneous advances in computing infrastructure and data gathering, significant strides have been made towards this goal over the last few years, now computers are capable of classifying images into different subcategories, libeling each pixel of an image and predicting the human pose with real time performances. However, most of this progress consists of assigning few labels to a set of elements of an image, and to obtain a visual understanding of the world around us through the lenses of a camera, we must leap forward and take it a step further.

In this work we address the problem of detecting and recognizing Human-Interactions in still images, Our two main contribution are, a state-of-the dataset for Human interaction, and a model capable of doing joint object detection and interaction recognition.