UNIVERSITÀ DEGLI STUDI DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE

DOCTORAL THESIS

Continual Object Learning

Author:

Luca Erculiani

Supervisors:

Prof. Andrea Passerini

Prof. Fausto Giunchiglia

A thesis submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

May 27, 2021

Abstract

This work focuses on building frameworks to strengthen the relation between human and machine learning. This is achieved by proposing a new category of algorithms and a new theory to formalize the perception and categorization of objects. For what concerns the algorithmic part, we developed a series of procedures to perform Interactive Continuous Open World learning from the point of view of a single user. As for humans, the input of the algorithms are continuous streams of visual information (sequences of frames), that enable the extraction of richer representations by exploiting the persistence of the same object in the input data. Our approaches are able to incrementally learn and recognize collections of objects, starting from *zero* knowledge, and organizing them in a hierarchy that follows the will of the user. We then present a novel Knowledge Representation theory that formalizes the property of our setting and enables the learning over it. The theory is based on the notion of separating the visual representation of objects from the semantic meaning associated with them. This distinction enables to treat both instances and classes of objects as being elements of the same kind, as well as allowing for dynamically rearranging objects according to the needs of the user. The whole framework is gradually introduced through the entire thesis and is coupled with an extensive series of experiments to demonstrate its working principles. The experiments focus also on demonstrating the role of a developmental learning policy, in which new objects are regularly introduced, enabling both an increase in recognition performance while reducing the amount of supervision provided by the user.

Acknowledgements

The pursuit of this doctorate have been by far the greatest adventure of my life, at least so far. As any legitimate adventure, it has been full of unexpected encounters, dramatic moments and victorious comebacks. Among the many characters that had a role in my PhD, I would like to start by thanking who have been present from the start, my advisors prof. Andrea Passerini and prof. Fausto Giunchiglia, and whom I cooperated with at the very end, the reviewers and committee members prof. Elisa Serafini, dr. Marco Schorlemmer and dr. Luciano Serafini. You give prestige to my PhD. In between these two extremes, I feel the need to thanks all the townfolks, ladies, dragons and spirits that populate this university. I had the pleasure to meet and know many of them during the course of the last three years. All of them had their role in my adventure. The work, support and technical advices I received was fundamental in enabling me to reach this ending. You have my eternal gratitude. At last I must thank everyone who was with me before the prologue and will be after the epilogue. These are the people outside the academia who cared for me and had the challenging role of comforting and helping me in overcoming difficulties of a world they were not accustomed to: my friends, my family, my love. This PhD is dedicated to you.

Contents

Chapter 1

Motivations

We are currently living in the golden age of Machine Learning. This field, after decades spent growing mainly inside academia, is delivering its power to the industry in an ever increasing number of sectors. At the same time, the application of techniques and algorithms developed in the field of Machine Learning enabled to boost the research of numerous fields, from Biology to Engineering, and to extract new knowledge from their existing data sources.

Among the many subfields of Machine Learning that experienced this dramatic explosion, one of the most prominent is Deep Learning and the theory of Neural Networks in general. This Reinassance of Neural Networks, whose ground theory dates back to the middle of the past century, originates from the encounter of the demand and offer; the latter is the availability of new solutions to accelerate the computations of these kind of architectures, while the former being a long list of real world tasks in which Neural Network excels, like text recognition and speech and image processing.

If we focus on Computer Vision, we can appreciate the disruptive innovation Neural Network fostered in the spotlight task of this field, which is image classification. The first Deep Learning architecture specifically developed for image classification achieved, at the time of its of introduction and after many years of incremental improvements, a reduction in the error rate of almost 20%, compared with the previous state of the art (a shallow

model) [46]. From that point on, deep architecture monopolized the charts of the majority of Computer Vision challenges.

The success of Deep Neural Networks in this field was made possible by the availability of huge training datasets. Deep models are famous for requiring huge collections of examples in order to be effective, thus the need of application specific accelerators. The bigger the data collection, the deeper the models can be, the better the performances they can obtain. This drove the whole field to produce ever increasing datasets in order to accommodate the needs of the models [2]; what was then one of the key enabler of innovation in these fields now is seen as a minimum requirement. This ever increasing growth has drawbacks too. Aside from the exorbitant cost of model training (a single training for the famous BERT neural language model is estimated costing tens of thousands of dollars [86]), this kind of architectures are not able to easily expand their knowledge, instead they require a new training from scratch each and every time new data is published (the so called *catastrophic forgetting*).

In a push to overcome the limitations of "mainstream" deep models, over the last few years many new tasks and challenges arose, many of them focused on fostering new ways to train models in constrained scenarios. We thus assisted to the genesis of new algorithms capable of obtaining performances comparable to full-fledged state-of-the-art algorithms of few years back, but requiring just a fraction of parameters (and thus computational power [36]). Other tasks focused on explicitly limiting the size of the training dataset (the so-called few shots and meta learning tasks). Others tackled instead the challenge of continuously and non disruptively adding new classes to the pool recognized by the models or rejecting the unknown classes.

All the new interest towards augmenting the flexibility of the models clearly depicts what is considered the direction that must be pursued, in order to allow for a new evolutive step for the field. It is not a coincidence that many of the developed techniques try to mimic

the way in which (allegedly) another kind of agents learn and reason: humans. Humans are masters of continuously learning new information, in transferring their knowledge, applying it in other domains and to reason in the Open World.

Multiple factors block machines to acquire the same learning abilities as humans, the most obvious being the fact that the biological mechanisms that govern learning are still obscure. Another difference between humans and machines is the interaction with the information and how in general the knowledge is provided. The information that is used to train modern machine learning models is organized following the principles, developed in the field of Knowledge Representation, of Ontology Engineering. In such structures the knowledge is articulated in a network of classes, organized by relations among them. A class in this context is nothing more than a set of individuals. In the vision realm, these are represented by sets of images. Humans on the other side learn from a constant stream of visual stimuli. Each thing that is perceived does not come with a string attached pointed to a node in a hierarchy of concepts.

This gap between what knowledge is for the machines (and what it is not for humans) fostered the proposal of new biologically inspired theories to organize information. Among them, Teleosemantics [55] proposes to view concepts as functions, and the knowledge of the machine as a set of abilities to perform specific functions. Without leaving the visual sphere, part of the concepts can be characterized as abilities to recognize specific types of perceived items. Subsequent works then expanded the theory, in order to build structures that are the counterpart of the standard Ontologies [29].

The focus of this thesis is to further develop settings and techniques to increase the similarity between the learning environment of the machine and the one of its user, combining together multiple characteristics of the constrained frameworks that made their appearance over the last few years. At the same time we wanted to explore new forms of interaction (and supervision) of the user with the machine, in order to increase the cooperation between

them. The next section will offer a brief overview of all the topics we covered.

1.1 Research issues and contributions

Reason and recognize the unknown with scarce data

The new interest towards settings like Few-Shot or Open World learning shifted part of the focus of the academic community towards less monolithic models, capable of reason in more dynamic environments. In Chapter 3 we present a framework that merges together the characteristics of these two settings while adding further constraints towards the goal of continuously recognizing both unknown and known objects, after the first encounter. We then present a first model capable of learning and recognizing instances of objects in this new setting. The model is capable of recognizing and adding unseen objects by itself, and to actively elicit a supervision from the user when necessary. This work was published at the 24th European Conference of Artificial Intelligence [19] .

From instances and classes to Objects

Chapter 3 presents a first, limited, approach towards the goal of recognizing any object. In order to extend the recognition capabilities from just instances to whole classes, Chapter 4 exposes a framework we developed a that enables to treat both instances and classes as being Objects of the same type. This was done by leveraging the assumption of a many-to-many correspondence between Objects and their representations (that comes from visual inputs). This is in contrast to the standard Ontology based theory, that usually uses a one-to-may approach where each example is associated to a unique class. This theory is grounded on the work of [29], and is based on a new type of interaction with the user, which is called to answer to a series of queries over the visual appearance of couples of Objects, rather

than giving a label to each of them. This work is at the time of writing under submission at Spinger Nature Computer Science journal.

Learning hierarchies of Objects

Chapter 5 details the last major contribution in this thesis, where the combination of the work presented in Chapters 3 and 4 is used to produce an algorithm capable of learning a true multi-level hierarchy of concepts bound to the requirements of the user that interacts with it. The agent is capable of estimating when to elicit the supervision of the user and to recognize new unseen objects and to add them inside its ever growing hierarchy of Objects. Via an interaction procedure that follows the principles detailed in Chapter 4, the user can intervene to guide the growth of the hierarchy following her own desires for what concerns the structure of the tree of Objects. This work is under submission to the Artificial Intelligence Journal.

1.2 Outline

The remainder of this thesis is organized as follows: Chapter 2 introduces various lines of research found in literature that are affine to our work. Chapter 3 details an algorithm capable of continuously recognizing and learning seen and unseen instances of objects. Chapter 4 lays the ground theory to extend the model in order to enable it to build, aided by the user, representations of groups of Objects that can be used to expand the recognition beyond single instances of Objects. Chapter 5 presents a first algorithm that takes full advantage of the theory presented in the previous Chapter, and that is able to learn and recognize full-fledged hierarchy of Objects. Finally, Chapter 6 summarizes all the work done, as well as foreseeing future developments and research directions.

1.3 Availability

Each algorithm presented in this work is tied to a python implementation and a dataset that are meant to be publicly released on acceptance of the corresponding paper. The code and dataset for the algorithms in Chapter 3 (accepted at ECAI 2020) were made available upon publication [1]. We also provide early releases of the code and datasets used in the works described in Chapter 4 and submitted to SNCS [2], as well as the one presented in Chapter 5 and submitted to AIJ [3].

[1] Availabe at https://github.com/lucaerculiani/ecai20-continual-egocentric-object-recognition
[2] Early release: https://github.com/lucaerculiani/towards-visual-semantics
[3] Early release: https://github.com/lucaerculiani/hierarchical-objects-learning

Chapter 2

Related Work

As stated in the previous chapter, over the last few years Deep Neural Networks led to massive improvements for the tasks of object detection and recognition in images [32, 77]. The main application scenario for this work is the use of large sets of photos, most of the time collected from the Web, to reliably identify objects from an increasingly large but static hierarchy of classes [81]. One of the key factors of the success of these approaches has been the availability of large datasets of (annotated) images and videos [14, 2]. A major disadvantage of these algorithms is the lack of adaptability and the huge amounts of resources they require.

Before introducing further details on the setting we propose, in this chapter we introduce the main techniques found in literature regarding the topics covered in this thesis. Furthermore we present a high level view of the learning mechanisms in humans. The chapter ends with an introduction to Teleosemantics, a Knowledge Representation theory that tries to overcome the classical dichotomy classes/instances and proposes to organize concepts similarly on how is done by humans.

The rest of the Chapter is structured as follows. Section 2.1 offers a brief excursus on the theories that were developed to model the learning process in the biological and cognitive field. Section 2.2 details the current progress in the field of Continuous and Open World

Learning. Section 2.3 summarizes the modern approaches to learn in contexts where scarce data is available. Section 2.4 presents the most relevant algorithms capable of dealing with the task of hierarchical classification. The chapter ends with Section 2.5, where the basis of the Teleosemantics theory are introduced.

2.1 Human learning

The themes of catastrophic forgetting and continual lifelong learning are known and studied in biology as well. In this field, the struggle to balance the effort to integrate new information without interfering with the consolidated knowledge is usually referred as the stability-plasticity dilemma.

Humans, and animals in general, excel in the task of continual lifelong learning. Our capacity to continuously learn and refine the knowledge over long periods, by interacting with the environment, is mediated by a complex set of cognitive and neurochemical abilities. These functions guide us in the early-age development and mediate the specialization of our perceptual abilities and motor skills [104].

Even if in the course of our life we tend to forget part of what we learned, it is rare for the new knowledge to directly interfere with the previously retained information. For example, a human can learn new motor skills without compromising the stability of the previously acquired ones [9].

Over the last fifty years many theories were proposed to explain the mechanisms that govern the learning inside the human brain. Among these, the two most popular ones are the Hebbian Plasticity and Stability theory [33] and the Complementary Learning System [57]. The Hebbian theory explains the stability-plasticity postulating that when a neuron drives the activation of another neuron, the strength of the link between them increases. During development, this effect drives the synaptic configuration towards a shape that offers

the optimal pattern of neural connectivity. In order to ensure stability, this system is then complemented with a controller that regulates the growth of the feedback between affine neurons.

The Complementary Learning System (CLS) [57] is based on the idea that different areas of the brain learns at different rates. Some areas show a fast adaptation and are responsible for the short-term memory. The information in these areas is encoded with sparse representations to minimize interferences. The short-term information is played back to other systems, that are responsible for the long-term storage. There, the information is encoded in overlapping representations to facilitate abstraction.

To date, there is no universally accepted model of how the learning works, at biological level, for humans. Nevertheless the theories introduced in these paragraphs have inspired a series of architectures that will be discussed in the next sections.

2.2 Continuous Learning

Incorporating novel classes is a notoriously hard problem for deep networks. The way in which these networks are trained drives them to learn models that implicitly follow the closed world assumption, and trying to dynamically expand their capabilities negatively affects previous knowledge. To overcome this, the task of continuous lifelong learning has been developed, that paved the way towards the Open World recognition task.

2.2.1 Lifelong learning

The task of continual lifelong learning is usually defined as a sequence of T study-sessions. In each study session t, the agent is provided with a batch of N_t labeled examples $\{(\boldsymbol{x}_{tj}, y_{tj})\}_{j=1}^{N_t}$, where $\boldsymbol{x}_{tj} \in \mathbb{R}^d$ is the d-dimensional vector containing the example, and y_{tj} is the corresponding label. The number of examples in every session may vary, and there is no *iid.*

assumption among the examples inside the same batch. Each session is considered a separate learning stage. In each stage, the agent is exposed only to the corresponding batch of examples. In spite of these limitations, the agent is allowed to store examples of previous sessions and use them in each learning stage. In each session t, a new classes are introduced.

The description of the setting matches the natural idea of an incremental learning scenario. The new information is gradually presented to the agent, which, ideally, must be able to build a representation for the new class that does not interfere with the ones already learnt without a complete retrain, so without building a brand new set of representations from scratch for all the other classes. Given that building class representations respecting this truly incremental setting has proven to be quite a hard task, the constraints of the scenario are relaxed, giving the ability to the agents to explicitly store information of the past sessions and combine them with the current batch of examples. The reason behind this grant is that the knowledge learnt from previous sessions could require small adjustments in order to fit the new one without major interferences. This approach can be seen as a middle way between pure continual lifelong learning and standard offline supervised learning.

A number of datasets were proposed to evaluate and compare different architectures. The most used datasets are built on top of famous datasets for vision or audio classification. The evaluation is done at the end of every training session. The results at the end of each session are then collected and organized in order to measure the overall capabilities while increasing the number of classes, as well as the performances of the models over the standard classification (meaning the results at the end of the first training session) and the accuracy over only the newly introduced classes at each session. Due to the nature of the task, these models usually imply some sort of dynamic mechanism to scale the resources they use during training, such as the number of neurons of the network. For this reason, the models are often compared in terms of their requirements in order to scale up the number of classes they handle. The same is done if the agents need to store and use examples from

previous sessions.

Among all the lifelong learning algorithms it is possible to identify two main approaches in the literature. The first approach is focused on limiting the interference between previous and new knowledge by imposing constraints on updates of the weights or other forms of regularization. Examples of this approach include estimating the importance of each individual synapses for a certain task in order to reduce the rate for the most relevant ones [105]. Similar methods can be complemented with custom loss functions that explicitly account for this behavior [42]. Many other methods deal with the new knowledge by allocating new neurons. Then the old weights can be completely frozen [82], or selectively retrained [103, 90, 18, 106]. Another approach makes use of a technique commonly referred as Knowledge Distillation [34]. It is based on the idea of transferring knowledge from a Neural Network to another by training the second to produce the same output distribution of the first one (in the case of the original work, the vector produced by the final softmax operation). This technique was adopted in Continual Learning mainly by training the two network on separate tasks, and then distilling the knowledge of one of them in the other [51, 99, 85, 38, 15, 59]

Another common technique that is used in the field is inspired by the Complementary Learning System theory, described in Section 2.1. These solutions are equipped with two different subsystems, in order to handle short and long term memory separately. The role of the two components is fundamentally different in different architectures. Some use the long-term memory module as a generator to sample in order to obtain pseudo-examples of the classes sees in previous sessions [88, 16, 4]. Others make use of both modules for classification, having a short-term classifier that exploits basic nearest neighbor techniques to predict new classes, while the long-term module is used to classify the old ones. A third module is used to decide which memory module should be used for classifying a new sample. The information from the short-term memory is then moved to the long-term

one by retraining, combining the content of the short-memory with past examples [25], or pseudo-examples [39, 40].

Albeit allowing for a non-destructive increase of the knowledge base, these methods still require a Closed World assumption. None of them is able to recognize the new objects as coming from a new class, requiring explicit supervision of the user any time new data appears. To overcome these limitations new solution have been developed, which will be presented in the next section.

2.2.2 Open Set and Open World

Getting rid of the closed-world assumption is a recent research trend in the machine learning community. The problem was first framed in terms of *open set* learning, i.e. learn to identify examples of unknown classes [84, 5]. More recently, *open world* learning has been introduced, where the learner should both identify examples of unknown classes, and be able to add them in the future as a novel class, if new external supervision is made available. As happens for most few-shot learning strategies, existing approaches for open world learning are similarity-based [6, 79]. Still, these works assume that novel classes are incorporated via class-specific training sessions, and their main objective is minimizing the effort required to update their internal representations. Our solution adapts this similarity-based principle to deal with user supervision in an online active learning fashion, and to work in an instance-level object recognition scenario.

2.3 Learning from few data

The perpetual addition of new knowledge and classes is not the only requirements towards a really developmental learning approach. All the framework cited in the previous sections

still require batches of examples in order to model a single class. Humans by contrast are able to model a new object by requiring very few encounters with it.

The desire of alleviating the requirements of huge training dataset led to the formulation of the few-shot learning task. Moreover, the requirement of reasoning with few examples has fostered the research in other fields too, where the nature of the setting imposed similar constraints. Examples are the field of human re-identification and human-guided interactive robotics.

2.3.1 Few-shots and meta learning

The basic idea of few-shots learning is quite straightforward: learn a task in a setting where very few data is available for training. This limitations seems not to fit with the requirements that modern deep learning architectures have, for instance, when dealing with the task of image classification [14, 81, 2]. For this reason, few-shot learning is tightly related to another research topic, called meta-learning. Meta-learning is the task of learning the learning process. In this scenario, the goal of the agents is to learn how to optimize the task of fitting a model for a specific problem. Given the scarcity of training data, few-shots learning has become a test bed for meta-learning algorithms.

Commonly used evaluation schemes [83, 76] have a quite similar structure. The training set is a collection of task sets $D_{meta-train} = \{D_t\}_{t=1}^{T}$. Each task set contains two subsets: a train batch of labeled examples $\{(x_{tj}, y_{tj})\}_{j=1}^{N}$ where $x_{tj} \in \mathbb{R}^d$ is the d-dimensional vector containing the example, and y_{tj} is the corresponding label. Each batch contains a fixed number m of classes, and for every class there are a number of examples n such that $nm = N$. Commonly used values for n are 1 and 5. The second batch (the test batch) contains other examples labeled with the same classes of the train batch. Each task set has to be considered as a different learning problem, virtually independent from the others. For this reason, different task sets contain different labels, that are shuffled in order to discourage

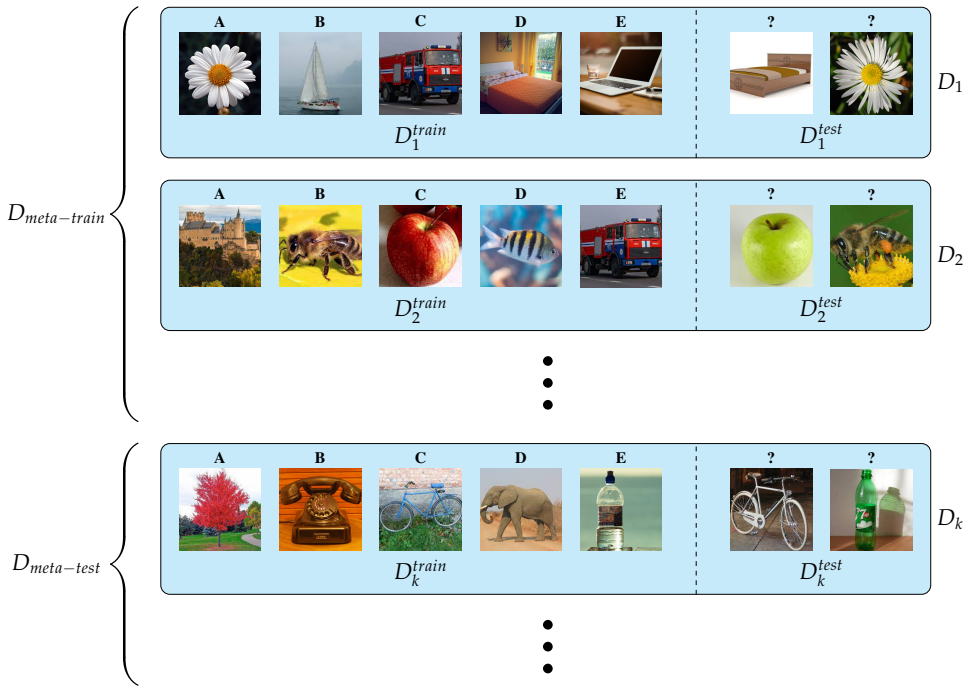


FIGURE 2.1: Representation of the data structures used in the task of Few Shots/Meta Learning. The train data $D_{meta-train}$ is composed by a set of datasets $\{D_t\}_{t=1}^{T}$, each of them containing the information for a different classification task. For each task, a fixed number of classes and examples is given (5 classes and one example per class in this case). Each task also contain a set of test examples to classify. Among different tasks there is no correspondence in the classes to predict. The test $D_{meta-test}$ and the validation $D_{meta-val}$ datasets (the second is omitted in the figure) have the same structure of $D_{meta-train}$.

to learn sample-class bindings. Two other datasets $D_{meta-val}$ and $D_{meta-test}$ are provided, with the same structure of $D_{meta-train}$, that are used for validation and final evaluation of the algorithms. During training the agent repeatedly fits a model for the task t using the train batch of the task set D_t, and then evaluates the performances over the test batch. The validation and testing is done following the same procedure over the tasks sets of $D_{meta-val}$ and $D_{meta-test}$.

Although works on meta-learning using deep networks have been circulating for quite some time [68], the topic gained serious traction only over the last few years. Many of the proposed architectures have dealt with the task via a metric learning approach. Some

of these method explicitly trained Siamese architectures to produce class representatives that are used to match examples in the test batches [44, 94, 95]. Other works made use of techniques commonly used in sequence labelling, treating the training batch as a sequence of elements coupled with their labels, followed by an element with no label (coming from the test batch), asking the network to predict the label for this element. Both recurrent [96, 76] and convolutional [67] architectures were presented. All of these approaches have in common the fact that they use a single global model to compare the data from train and test batches. Another approach, instead, keeps a separate global state that is used to initialize task specific classifiers [21]. These classifiers are then finetuned with the task specific examples. The error on the test set is then used to update the global state.

2.3.2 Human re-identification and human-aided robotics

The task of discriminating objects at instance level is usually referred as *re-identification*. In this setting the system is provided with N collections of input data. Each of these collections contains visual information (one or more frames of a video, for example) of a particular instance of an object, often referred as *identity*, different from all the others. Then the algorithm is provided with another set of M collections of data, each of them having one of the identities of the initial collection. The task of the algorithm is, for each of the new collections, to rank the original N identities by similarity. Then the performances of the algorithm are assessed by computing the Cumulative Matching Score (CMS), which represent the fraction of the M identities that had their corresponding identity in their rank up to position k.

No work tackling the re-identification task for generic objects was found in literature. The most prominent applications in this field are related to the identification of humans, for both static images [102, 50] and video streams [97, 58, 100]. Most of the approaches rely on Siamese Neural Networks, trained to embed images or videos of individuals into

vectors that are then compared by using various forms of similarity or distance metrics. Part of these methods rely on submodules specifically tailored to deal with human recognition, such as components that search for parts of the human body [102]. Similar approaches were applied for the task of vehicle re-identification from surveillance cameras [52, 87]. The main drawbacks of these approaches is the fact that they were developed to deal with single categories of objects.

In the robotics field, a number of works have focused on studying approaches for human-guided interactive learning [98, 43, 91, 69, 72]. In this setting a machine is trained to recognize objects that are manipulated by a human [98, 43], or directly by the robot [69], asking supervision to the user if the object is considered unknown. While sharing similarities with our setting, these approaches are focused on recognizing what is known and rejecting what is unknown, and always require human intervention in order to expand the pool of known objects. In contrast, we aim at building systems that can autonomously discriminate the unknown and integrate it with the previous knowledge, expanding the pool of recognized entities even without human intervention.

2.4 Hierarchical learning

The idea of framing the learning task around a setting in which the data is inherently organized in a hierarchy is common to multiple domains. Among all, the prominent ones in the literature concern the task of text classification and the one of gene/protein classification. For what concerns text classification, one of the first approaches, proposed by [45], aimed at exploiting the hierarchical arrangement of words in order to decompose the global classification task (in their case assigning topics to fragments of text), in multiple local classifications, each aimed at discriminating between the children of a particular node in their predefined hierarchy. The final prediction was then the last of a series of predictions starting

from the root of the tree. Each of the classifiers was implemented as a Bayesian model. Soon after [45], other works proposed the use of SVMs [17] or tree-structured Neural Networks [80] to accomplish the same task. [20] proposed techniques derived from boosting methods. With the advent of the deep learning era, these techniques were surpassed by sophisticate Deep Networks [37, 48], and were mainly abandoned.

In the realm of Bioinformatics, the proteins (and thus the genes that encode them) are usually organized in a hierarchy based on their functions. Multiple representations were proposed and made available to researchers, such as the Gene Ontology Project [3] or the Enzyme Commission classification system. These tree-like ontologies in turn enabled the development of a multitude of works based on the task of classification of proteins [73, 41, 74, 8, 11]

The use of hierarchical approaches is not limited to the fields cited in the previous paragraphs. The organization of knowledge in tree-like structures is a pivotal feature of many supervised and unsupervised learning algorithms. In [47] the authors exploited the knowledge of an Ontology to organize a large number of classes of objects in a tree, and then trained a separate neural network for each level of the hierarchy, combining their output at test time in order to boost the accuracy. Another example is the field of hierarchical clustering [13], an unsupervised setting in which the models are expected to return a hierarchy in which the leaves are the single clusters. These unsupervised techniques can be then combined with supervised learning algorithms. [107] made use of hierarchical clustering, in place of an Ontology, coupled with a Neural Network to boost the recognition of a large number of classes of objects.

The field of structured output prediction often deals with tree-like structures. The main difference in this context is the fact that the trees are the output rather than the input. A notable example is the task of phrase structure parsing, where the classifier receives as input a phrase and must output its syntactic parse tree [1].

Our goal differs from all these works. We are interested in learning and predicting in a setting where the objects belong to specific nodes in a hierarchy, but this hierarchy is dynamic and, starting from zero nodes, grows indefinitely following the desires of the user.

2.5 Knowledge Representation

In our view on of the root cause of the limitations that all the previous work have lies in the model used to represent and interact with the knowledge. The mainstream way to represent the knowledge is based on the idea that concepts can be organized by means of instances and classes. The former represent sets of individual elements, while the latter encode properties that are used to organize and group the instances. This way of representing the knowledge is usually referred in philosophical literature with the name of *Descriptionism* [65]. According to this theory, each concept is a representation of something, in terms of its properties. Artificial Intelligence and knowledge Representation were hugely influenced by Descriptionism, mainly because it is natural to organize knowledge via classification. In the context of this theory objects are *endurants*, meaning that they are wholly present at any given moment of time, always satisfying the properties defined by their classes.

An alternative approach is the one provided by *Teleosemantics* [55]. This school of thought provides an interpretation of *concepts* in terms of *functions*, *i.e.* abilities to perform specific tasks. For what concerns this thesis, the most influential work was the one developed by the philosopher Ruth Millikan [64, 61, 65, 63], further formalized by Giunchiglia, Fumagalli and Bella [29, 22]. The work of Millikan is focused on the process of recognition, and it is built starting from what she calls *substance concepts*, which represent abilities of recognizing certain types of items (that in the Teleosemantics theory are referred as *substances*). Anything that a human is interested (and capable) to recognize is bound to a substance concept. A substance concept can be seen as some sort of apparatus in the human

brain that is devoted to recognition of a specific thing (the substance), and whose goal is to accumulate the information, via experience, needed to perform this task. The experience required by a certain substance concept is obtained by being exposed to the substance, a process that is usually defined as *encounter*. The information contained in each encounter can be used to update the representation modeled by the substance concept. The fact that this process of refinement needs to happen implies that in this context the objects are *perdurants*, meaning that it must be assumed that a full (visual) picture of the items is never available but that their representation is built progressively, in time, as it is always the case in our everyday life.

The fact that perdurants are never experienced in a complete manner pose some challenges for the recognition process. How we can recognize something that, due to its nature, does not always present a set of uniquely identifying properties? Millikan gives us a solution: in her theory perdurants are those things (quote from [65]), "... about which you can learn from one encounter something of what to expect on other encounters, where this is no accident but the result of a real connection ...". Thus perdurants are assumed to have some characteristics that *persist* over multiple encounters. The *persistency* of the perdurants is a key property of perdurants, which will be exploited multiple times in this thesis. The persistency of objects is both *spatial* and *temporal*. The spatial persistency refers to the assumption that small spatial changes of the objects will result in small variation of their appearance. For instance a cup, if rotated by a minuscule angle, will still preserve the appearance it had before the rotation. Thus the two cups (the one before and the one after the rotation), can be easily recognized as being the same object. The temporal persistency refers tho the tendency of most of the objects to retain their appearance over time, thus easing their recognition in different encounters. This property is not absolute and can even vary depending on the time between different encounters. For instance a person would be able to recognize her best friend the day after their last encounter, but if they are kept separated for

twenty years, the changes in the visual appearance of her pal could deny the recognition.

Another consequence of the nature of perdurants regards the nature of the substance concepts (their mental representations) they generate. The relation between substances and substance concept is many-to-many. Multiple substances can be recognized by the same substance concept. For instance, an inexperienced person could be unable to distinguish different species of felines, thus recognizing all of them as generic cats. This is similar to the relation between classes and instances used by descriptionist models (which implies just a one-to-many relation). The key difference of Teleosmantics is the fact that the same substance can be recognized by multiple substance concepts. In a person will struggle to re-identify her best friend if seen from a long distance or in the dark, and will probably recognize him as a generic guy. Thus, depending on its appearance, a substance can be associated to different substance concepts. This must not be considered as the equivalent, for Descriptionism, of a classification error; the recognition from the point of view of that particular person was correct, but that encounter did not contain enough information to further recognize the substance as her pal. As a result of this formulation, the result of the recognition process is highly influenced by both the subject (different humans have different substance concepts) and the context (different encounter will be associated to different substance concepts).

Substance concepts play no role in organization of knowledge. They sole purpose is recognition, they play no role in the semantic interpretation of the knowledge. The ability to organize the knowledge is provided by the *classification concepts*. These concepts model the abilities "... of simplifying the environment, of reducing the load on memory, and of helping us to store and retrieve information efficiently ... " [65], and mediate reasoning. The classification concepts are responsible, as the name suggests, for the classification of objects, but they do not perform the recognition of visual stimuli. The mapping of substance concepts onto classification concepts enable humans to link recognition and classification.

Chapter 3

Continual Egocentric Learning

We are interested in recognizing objects in a setting which resembles the one under which humans see and perceive the world. This problem is of high relevance in all those applications where there is a wearable camera (e.g., the glasses) which generates images or videos whose recognition can be used to support the user in her local needs (e.g., everyday life, working tasks). The main innovative characteristics in this setting are: (i) it assumes an egocentric setting [93], where the input is the point-of-view, and the desires, of a single person (i.e., the data has low diversity and high correlation); (ii) there is a continuous flow of input data, with new objects appearing all the time (i.e., we assume the agent operates in an open world); (iii) recognition should be able to go as deep as instance-level re-identification (e.g. recognizing my own mug); (iv) supervision is scarce and should be solicited to the user when needed, also accounting for entirely autonomous identification and processing of new objects.

This scenario contrasts with the typical setting in which deep learning architectures shine. Incorporating novel classes is a notoriously hard problem for deep networks. The way in which these networks are trained drives them to learn models that implicitly follow the closed world assumption, and trying to dynamically expand their capabilities negatively affects previous knowledge (the so-called *catastrophic forgetting* [30]). While substantial

progresses have been made in fields such as continual lifelong learning [75] and few-shot learning [44], state-of-the-art algorithms in this field are far from being able to match the capabilities of humans. We argue that a key factor for this gap is the way these algorithms are exposed to the data during training, with respect to what happens for humans and other animals. Humans experience the world via a continuous stream of highly correlated visual stimuli, initially focused on very few objects. This enables them to progressively acquire an understanding of the different ways in which objects can appear, and on the similarities and differences between objects.

On these premises, the solution we developed is based on the following intuitions:

- Introduce persistency (as defined in Section 2.5) as a key element for instance-level labeling. By this we mean the fact that when we see an object as part of a video the object will change very slowly, allowing to identify *visual invariances* useful for sub-sequence recognition. This is thought to be one of the key aspects for early visual learning in children [23, 71].

- Use similarity as the main driving principle for object recognition and novelty detection. This is consistent with the recent trend in few-shot [44, 94] and Open World [6, 79]) learning, and we extend it here towards the autonomous recognition of new objects.

- Progressively introduce novel objects in a developmental fashion [7, 93], and provide supervision on-demand in an online active learning fashion.

The rest of the Chapter is structured as follows. The setting and the framework we developed are detailed in 3.1, while in Section 3.2 we present the result of an ample set of experiments we realized both on publicly available dataset as well as a dataset we collected, specifically meant for this task. Finally, in Section 3.3, we explicit some final remarks on

this work, as well as detailing the limitation of this approach (that are partially addressed in the next Chapters).

The work presented in this Chapter was published at the 24th European Conference on Artificial Intelligence [19].

3.1 The framework

The previous chapter highlighted the solutions found in literature that are the closest to our line of research. Still no work was found that is able to achieve our goal of recognizing any object it encounters (as a new or an already seen object), while dynamically add to the pool of recognized items each new object it encounters. In this section we will further detail our setting and provide a first straightforward algorithm that is able to tackle the task.

3.1.1 The setting

As explained at the beginning of the chapter, we focus on Open World, incremental instance-level object recognition, with an egocentric view bound to a specific user. Note that we assume maximal granularity, meaning that each object has its own class, and the goal of the algorithm is to cluster input data rather than assigning a label to each sample. In the following chapters we will relax this assumption, enabling the recognition of classes of objects.

The information is made available to the learning agent via a continuous stream of data. Each sample in this stream is a short video, focused on a certain object. The video simulates the point of view of an user who is interacting with the object via manipulation. The objects are thus small enough to be handled by hand, and are subjects to a series of rotations and deformations, which are performed in order to inspect/recognize the object and its functional use. The same object is encountered multiple times in different environments (*i.e*

with different backgrounds). The goal of the agent is, for each new video, to determine if it contains an object it has already seen, or if it shows an object it sees for the first time. After the first encounter, the agent should add the new object to the pool of objects it knows, so as to recognize it in the future. The agent can query the user in order to gather supervision on instances it considers uncertain and prevent taking wrong decisions, in an online active learning fashion.

As discussed in the previous chapter, although this setting is affine in many aspects to a number of other tasks, the combination of its characteristics make it extremely challenging. In particular the fact that the input data (the frames of the video sequences), albeit abundant, is highly correlated makes training any kind of model a difficult task. This is true even for algorithms developed for the task of Few Shots Learning (discussed in Section 2.3.1), that can still access a train time to a fair amount of examples. The best sounding approach is then to tackle our recognition task with a similarity based approach. In this perspective, the target of the algorithm is then to build representations of objects capable of extracting all the information contained in the input data. If built correctly, these representations can be then used to perform recognition and classification by computing their distance/similarity with the examples available at training time, even when the training data is scarce. The basic assumption of these techniques is that input frames containing the same objects will be closer together/more similar than frames containing different ones, *i.e.* that, in the features space, the classes of neighbors of the input objects are a good approximation of the class of the newcomer. This is the principle assumed by K-Nearest Neighbors (KNN), one of the simplest (and yet effective) Machine Learning algorithm, and it is in turn the approach chosen by many of the works we discussed in Chapter 2 tackling tasks like Open World learning, Meta Learning and Re-Identification.

In principle any technique that is capable of building a representations of input data is viable. In practice the constraints we discussed in the previous paragraph limit the range

of effective approaches. Limiting the input information to just uncorrelated examples (for instance selecting a single frame for each sequence) would make the training data too small. An alternative approach would be employing models specifically meant to deal with correlated (sequential) data, such as Recursive or unidimensional Convolutional Neural Networks. The drawbacks of these models is that they employ even more parameters than their equivalent approaches for non sequential data, thus requiring even more data. Moreover, recent works started questioning the competitive advantage of such architectures when dealing with visual classification [101, 54]. In the end we decided to employ part of a network pre-trained on a different task (image classification in this case) to obtain a representation of each single frame in the input sequences, a strategy that is quite common in settings in which there is no easy way of training an ad-hoc model. We then decided to further simplify our solution by aggregating the different representations of frames in a single one by averaging them. The choice of using the average as aggregations is also motivated by the environment in which the input data is captured. Due to the presence of a fair amount of potentially useless information (coming for instance form the background surrounding the target object), the average can help to retain the invariances in the sequence, the target object, while filtering part the noise and variance in the input data. This is obviously a tradeoff, due to the fact that the variance in the sequence could also come from some features of the object, but it adds some form of robustness against outliers.

There is another implication, for our setting, in choosing to use a network that was pre-trained for the task of classification. The set of classes to which the model was exposed on training could prevent it to distinguish single elements of a particular class. For instance a model trained to recognize the class "cup" could be unable to build useful representations of different cups. The reason of this is the fact that the training it received explicitly pushes the network to ignore intra-class invariances. This drawback can be in part avoided by using, after training, just the first layers of the network as embedding model, discarding the last

Algorithm 1 Open world egocentric object recognition

Input : a real value $\alpha \in [0,1]$

procedure FOLLOWER(α)
  $\mathcal{M} \leftarrow \emptyset$; $\mathcal{K} \leftarrow \emptyset$
  while $s \leftarrow$ NEXTVIDEO() **do**
    $v_s \leftarrow$ EMBEDVIDEO(s)
    $v_{min}, l_{min} \leftarrow$ GETNEARESTNEIGHBOUR($v_s, \mathcal{M}$)
    $\delta \leftarrow$ D(v_s, v_{min})
    $\lambda_l, \lambda_u \leftarrow$ GETDECISIONTHRESHOLDS($\mathcal{K}, \alpha$)
    if $\delta < \lambda_l$ **then**
      $l_s \leftarrow l_{min}$
    else if $\delta > \lambda_u$ **then**
      $l_s \leftarrow$ NEWID()
    else
      y $\leftarrow$ ASKUSERSUPERVISION(v_s, v_{min})
      if y **then**
        $l_s \leftarrow l_{min}$
      else
        $l_s \leftarrow$ NEWID()
      $\mathcal{K} \leftarrow \mathcal{K} \cup \{\langle \delta,$ y$\rangle\}$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\langle v_s, l_s \rangle\}$

ones that are more closely related to class discrimination. This is the approach we followed in our work; later in Section 3.2.1 we show the results of an experiment we did to confirm that the model we used still allows for intra-class discrimination.

3.1.2 The Algorithm

The pseudocode of the algorithm is shown in Algorithm 1. It takes as input a parameter $\alpha \in [0,1]$ that represents the amount of effort the user is expected to provide on average, in terms of number of queries. α must not be considered as an hyperparameter of the model due to the fact that is used to model a characteristic of the user, rather than a configuration of the algorithm we present. For this reason, in the experiments later in the chapter, α will not be optimized but used to explore the behavior of the framework at different levels of supervision.

The algorithm consists of an iterative routine that receives from the environment a new video s (containing a single object) at each iteration. By exploiting the notion of persistency

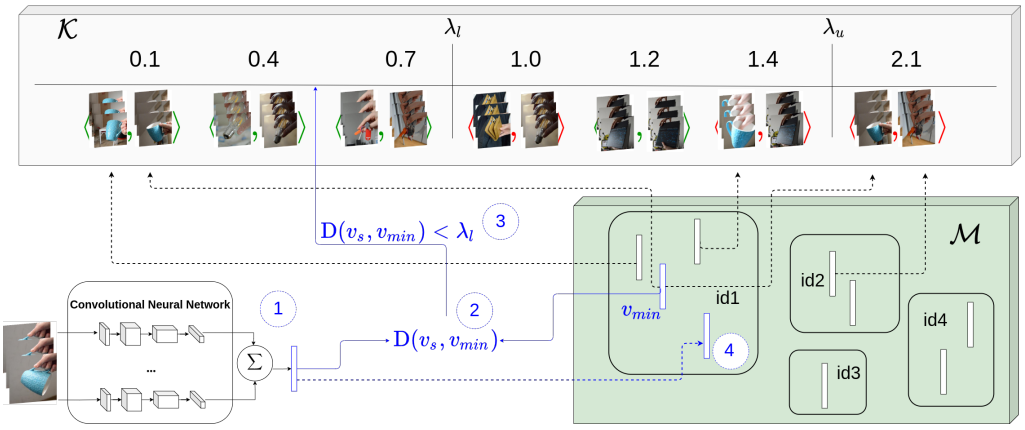


FIGURE 3.1: Graphical representation showing an iteration of our algorithm. In black the state of the algorithm at the end of the previous iteration. In green and red the couples of same and different objects inside $\mathcal{K}$. In blue the operations that occur at the current iteration. The new video s is converted in an embedding v_s (1) and its nearest neighbor v_{min} is identified (2). Given that their distance is lower than λ_l (3), v_s is added to the memory $\mathcal{M}$ with the same identifier of v_{min} (4), and the iteration finishes. No user feedback is required in this iteration.

as defined in Section 2.5, the algorithm assumes that the object in the video sequence is the same in all the frames. thus the information of each single frame can be combined in a single representation. The new video is transformed in a fixed size representation $v_s \in R^n$, by generating embeddings for each frame using ResNet152 [32], and computing the mean of the embeddings of the frames. We tested more advanced aggregation methods, such as LSTMs, but we found these underperforming, likely because these architectures require an offline training phase with a lot of training data, a rather different setting from the one we are facing.

The resulting representation is compared with a collection of representations of past videos, that the algorithm has stored in its memory. The algorithm then decides if there exists a representation in memory that contains the very same object, at instance level. It starts by computing the distance δ between the new representation and its nearest neighbor v_{min} (in memory). Based on this distance it needs to decide among three possible options:

1) the object is the same as the one retrieved from memory; 2) the object is a new one; 3) there is not enough evidence to make a decision, and the user feedback is needed. The choice among the three options is made by comparing the distance δ with a pair of thresholds λ_l, λ_u, computed using a procedure described later on. If $\delta < \lambda_l$, the object is recognized as already seen (option 1), and it is assigned the same identifier l_{min} of its nearest neighbour. If $\delta > \lambda_u$ the object is considered a new one (option 2) and it is assigned a brand new identifier. Otherwise, a user feedback is requested (option 3). In this latter case, the user is asked whether v_s and v_{min} are the same objects, and the identifier of the object is updated according to the user feedback. Finally the new object-label pair is added to the memory $\mathcal{M}$. If the algorithm decided to request the user supervision, the answer to its query is stored in another memory $\mathcal{K}$, together with the distance δ. Then the whole process is repeated with a new video.

As described above, the whole decision process depends on the values of the two thresholds λ_l and λ_u. These thresholds are estimated at each iteration by using the supervision provided by the user in previous iterations,which is used to build a collection $\mathcal{K} = \{\langle \delta_i, y_i \rangle \mid 1 < i < |\mathcal{K}|\}$ of distances between pairs of objects $\delta_i = \text{D}(r_i, r'_i)$, coupled with a boolean value y_i, that represents the supervision from the user (i.e., $y_i = \top$ if r_i and r'_i are the same object, $y_i = \bot$ otherwise). As explained in Section 3.1.1, the underlying assumption is that the embeddings of various instances of the same object are closer together than those of different objects. Based on this assumption, GETDECISIONTHRESHOLDS sets the two thresholds so as to maximize the probability that, if δ is inside these boundaries, the algorithm would take a wrong decision on the corresponding object, given the information currently stored in memory. This is achieved by solving the following optimization problem:

$$\underset{\lambda_l,\lambda_u}{\text{argmax}} \quad H(\mathcal{Y}_{\lambda_l}^{\lambda_u}) - H(\mathcal{Y}^{\lambda_l}) - H(\mathcal{Y}_{\lambda_u}) \tag{3.1}$$

$$\text{subject to:} \quad \mathcal{Y}_{\lambda_l}^{\lambda_u} = \{y|\langle\delta, y\rangle \in \mathcal{K} \wedge \lambda_l \leq \delta \leq \lambda_u\}$$

$$\mathcal{Y}^{\lambda_l} = \{y|\langle\delta, y\rangle \in \mathcal{K} \wedge \delta \leq \lambda_l\}$$

$$\mathcal{Y}_{\lambda_u} = \{y|\langle\delta, y\rangle \in \mathcal{K} \wedge \lambda_u \leq \delta\}$$

$$\sum_{i=1}^{|\mathcal{K}|} \mathbb{1}(\lambda_l \leq \delta_i \leq \lambda_u) = \lceil \alpha|\mathcal{K}| \rceil \tag{3.2}$$

Here $\mathcal{Y}_{\lambda_l}^{\lambda_u}$ is the subset of $\mathcal{K}$ of user answers related to objects with a distance within the two thresholds, while $\mathcal{Y}^{\lambda_l}$ and $\mathcal{Y}_{\lambda_u}$ are the subsets related to objects below λ_l and above λ_u respectively. The function H returns the entropy of a given set. $\mathbb{1}$ is the indicator function mapping true to 1 and false to 0. Eq. 3.2 imposes the constraint that the user is queried with probability α, where the probability is estimated using the currently available feedback. The objective function is chosen in order to set the two thresholds in the area where the algorithm is maximally confused, adjusting the size of the area to the effort the user is willing to provide. During training, the algorithm will receive supervision only for those examples where $\lambda_l \leq \delta_i \leq \lambda_u$. Thus only these elements will be added to the supervision memory $\mathcal{K}$, increasing the fraction of elements in $\mathcal{K}$ having a distance between the two thresholds. This, combined to the fact that α remains constant during training, will eventually lead to selecting two closer and closer values for λ_l, λ_u, reducing the size of the area of confusion and thus the probability to request supervision. An alternative approach could be leaving the algorithm the freedom to select the optimal size for the confusion area (leaving α as an upper limit). We tested this approach, but we found that this led to instability during training, due to poor estimates of the correct size of the confusion area.

Another advantage of formulating the constraints as in Eq. (3.2) is the ability to solve

the maximization problem efficiently, provided that the content of $\mathcal{K}$ is stored inside a list that is kept sorted with respect to δ_i. We refer to this list as $sorted(\mathcal{K})$. Then the optimal solution is associated to one of the contiguous sub-lists of $sorted(\mathcal{K})$ of length $\lceil \alpha |\mathcal{K}| \rceil$. Let $\langle \delta_j, \lambda_j \rangle$ and $\langle \delta_k, \lambda_k \rangle$ be the first and last element of a sub-list L, where j,k refer to the positions of the elements in the full list $sorted(\mathcal{K})$, so that $k = j + \lceil \alpha |\mathcal{K}| \rceil$. Setting $\lambda_l^L = \delta_j$ and $\lambda_u^L = \delta_k$ guarantees that the constraint in Eq. (3.2) is satisfied. Our algorithm evaluates the objective (3.1) for each of the contiguous sub-lists (there are at most $|\mathcal{K}|$ of them) and returns the values $\lambda_l = \lambda_l^{L^*}, \lambda_u = \lambda_u^{L^*}$ corresponding to the sub-list L^* for which the objective is maximized.

For evaluation purposes as well as to allow the algorithm to work in absence of user supervision, it is useful to have a modality where asking the user is not an option, and the algorithm can only decide between recognizing the object as already seen or considering it a new object. This can be done defining a single "recognition" threshold λ_r, such that if δ is lower than the threshold the object is recognized as the same as its nearest neighbor, otherwise it is considered a new object. The threshold can be set in order to maximize the number of correct predictions given the available supervision $\mathcal{K}$, by solving the following optimization problem:

$$\lambda_r = \underset{\lambda}{\operatorname{argmax}} \sum_{i=1}^{|\mathcal{K}|} \mathbb{1}((\delta_i < \lambda) \oplus \neg y_i) \tag{3.3}$$

where $\mathbb{1}$ is the indicator function mapping true to 1 and false to 0, and $\oplus$ is the exclusive OR. This problem too can be solved in time linear in the size of $\mathcal{K}$, by just testing all thresholds $\lambda^i = \frac{\delta_i + \delta_{i+1}}{2}$ for $i \in [0, |\mathcal{K}|]$ (where $\delta_0 = \delta_1 - \epsilon$ and $\delta_{|\mathcal{K}|+1} = \delta_{|\mathcal{K}|} + \epsilon$ for an arbitrary small ϵ).

Even this maximization problem can be solved efficiently by using the sorted version of $\mathcal{K}$. The solution is linked to one of the $|\mathcal{K}|$ ways to partition $sorted(\mathcal{K})$ in two contiguous

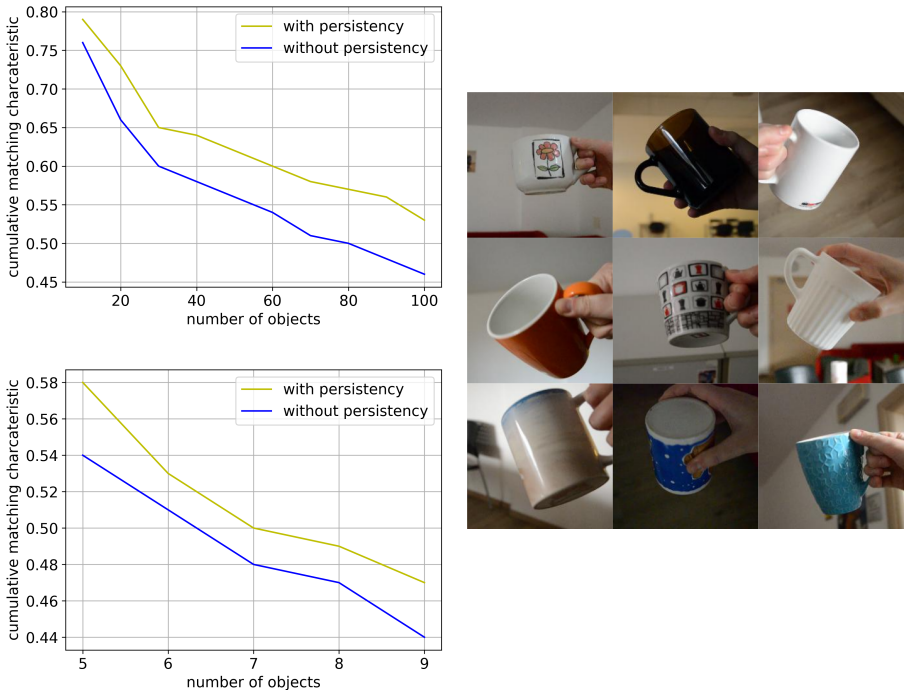


FIGURE 3.2: Re-identification results on generic objects (upper left) and on the coffee mugs (lower left) depicted on the right.

lists. For each index i in $sorted(\mathcal{K})$, we can partition the list in two sub lists containing indices where ranges are $0..i$ and $i+1..|\mathcal{K}|$, by selecting a threshold $\lambda^i = \frac{\delta_i + \delta_{i+1}}{2}$. In order to find the optimum, we evaluate equation 3.3 for all the $|\mathcal{K}|$ candidate thresholds λ^i.

3.2 Experiments

The framework described in Section 3.1 differs in many aspects from the ones of mainstream classification algorithms. For this reason, commonly used benchmarks meant for classification, even those employed for continuous lifelong learning, aren't suited to evaluate our approach. The ideal dataset should contain a collection of videos, each focused a single instance of an object, with the same object appearing in multiple videos over different

backgrounds, somehow emulating the first stages of developmental learning under an egocentric point-of-view [93]. The egocentric perspective was selected because it is the most natural setting for this task. It mimics the behavior of a user who focuses on an object she is interested in, rotates and deforms different perspectives.

As far as we know only two public datasets satisfy these requirements [78, 53]. The larger of the two, called CoRe50 [53] contains a total of 50 objects, each recorded 11 different times. Given that our goal is to measure the ability to deal with the unknown in an incremental scenario, we need to prioritize the number of different objects over the number of recorded sequences. Thus, we collected a new dataset of videos of objects commonly found in office/home spaces. The dataset contains 500 short videos of 100 different objects, 5 videos per object, recorded with different backgrounds, and it is freely available together with the python code used to run the experiments [1]. The main findings are however confirmed when evaluating our algorithm on the CoRe50 dataset.

3.2.1 Re-identification and the role of persistency

Our first experiment is aimed at investigating how exploiting persistency in space-time affects the recognition performance of our model. We compare our algorithm with an alternative model that does not make use of this feature. Without persistency, each frame of a sequence should be classified independently of the others, as if they were a collection of pictures. We performed these tests on a closed-world re-identification setting, similar to the one used in person re-identification [50, 100]. The aim of these tests is to evaluate the role of persistency independently of the other aspects of our framework, like Open World recognition and online active learning.

For each object in our dataset, we sampled one video for the training set and one for the test set. For each test sample, we then computed its nearest neighbor in the training set,

[1] Available at: https://github.com/lucaerculiani/ecai20-continual-egocentric-object-recognition

where samples are videos when using persistency, and individual frames otherwise. Figure 3.2(upper left) reports the fraction of videos (or frames) that have as nearest neighbor a sample of the same object (i.e., we report the Cumulative Matching Characteristic used in re-identification, with a number of positions $k = 1$), averaged over 100 folds, for an increasing number of objects to re-identify. The advantage of using persistency is apparent, and while increasing the number of objects clearly decreases the performance of both alternatives, the gap is substantially preserved.

The network we use for embedding frames (ResNet152) was originally trained to classify objects of 1000 different classes of ImageNet, a database that follows the same structure of the WordNet lexical database. Even if the original training set covered a broad spectrum of the objects commonly found in tasks recognition (including some in our own dataset), ResNet was trained with class-level supervision, a way that explicitly pushes the network towards suppressing intra-class variance.

In order to assess the performance of our algorithm at instance-level object recognition, we performed a second series of re-identification experiments focused on the "coffee mug" synset. Note that the synset appears as a leaf in the tree of ImageNet, i.e., the network we use for the embedding is not trained to distinguish between different mugs, but to consider them as a single category. We recorded a small collection of videos of nine different coffee mugs (shown on the right of Figure 3.2). Figure 3.2(lower left) presents the re-identification results. As expected, performance are lower than the ones for generic objects, but the persistency model still obtains consistently better results than the frame-based one.

3.2.2 Open World recognition

The next series of experiment aims at evaluating the performance of our algorithm in the setting described in Section 3.1.1. One of the key elements that distinguish our setting from a standard object recognition one is the fact that data arrive as a continuous stream, as typical

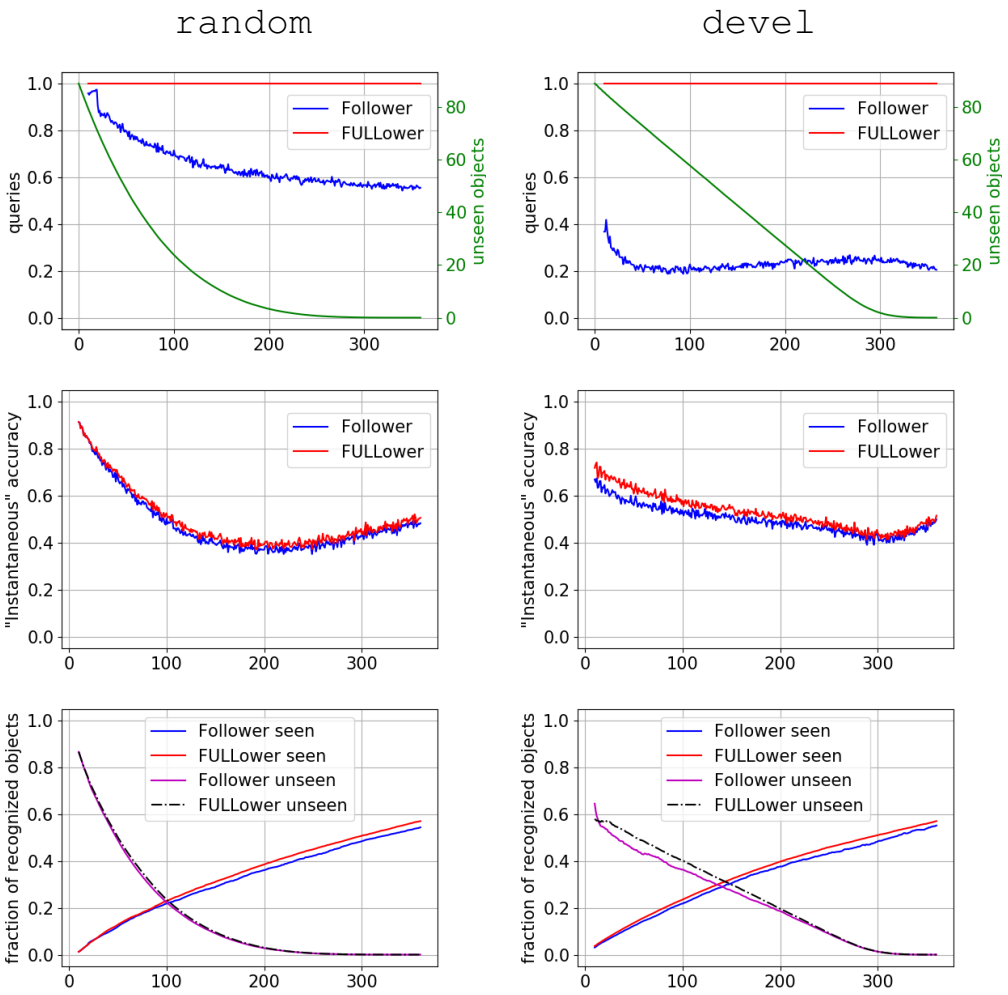


FIGURE 3.3: Open world recognition results for increasing number of iterations (on the horizontal axis).

of online learning settings, and the order in which data are presented to the algorithm can have an impact on the quality of learning [7]. In infants, the ratio at which novel objects appear is considered crucial for the development of their recognition capabilities [93, 12]. In order to investigate how this can affect our recognition algorithm, we considered two different policies for introducing novel objects. The fist one, named `random`, presents videos in a random order. The second, that we named `devel` (short for developmental), at each iteration picks the video of a randomly chosen unseen object with probability 0.3, and randomly selects the video of an already seen object otherwise. We chose the probability

0.3 to evenly spread the encounter of new objects. Lowering the probability would result in showing all the videos of already seen objects before introducing a new one. Raising the probability would eventually result in showing all the unseen objects before showing a video of an already seen object, resulting in a behavior similar to the one obtained showing the sequences in random order (or even more pronounced). While being less intuitive, the `devel` setting is less artificial than the `random` setting, especially when the number of objects increases. Assuming that the dataset contains each object in the world, a `random` setting would imply that the user sees the majority of objects once, before interacting with an already seen object. That is not how a human interacts with the environment and learns. In this sense, the `devel` setting is clearly a better approximation of the behavior of a human.

For each of the two policies, we compared our recognition algorithm, FOLLOWER (see Algorithm 1) with a fully supervised version of the algorithm (which we refer to as FULLOWER), where user feedback is requested at each iteration.

The dataset was divided in three subsets. First, 10 objects were randomly selected, and all their sequences (50 in total) were used to perform unsupervised evaluation after the online training phase. For each of the remaining 90 objects, four videos were used for the interaction with the algorithm, while the fifth was kept in a separate set for online evaluation in the training phase. The procedure was repeated 2000 times with different permutations of the data and results were averaged.

Supervised

As FOLLOWER needs as input the expected user effort α, we tuned it so as to ask the minimum supervision required to have similar performance as FULLOWER. For the `random` setting, this was achieved with $\alpha = 0.92$, while in the `devel` setting, $\alpha = 0.35$ was sufficient. For the first 10 iterations FOLLOWER was forced to always ask for supervision, in order to bootstrap the estimation of its internal thresholds.

Figure 3.3 shows the results of our experiments in terms of user effort and recognition quality for an increasing number of objects presented to the algorithms, for the `random` (left column) and the `devel` (right column) setting respectively. The upper row shows the average fraction of objects seen at least once up to that iteration included (green curve). The plot also shows the number of queries to the user for FOLLOWER (blue curve) and FULLOWER (constantly one, red curve); these can be interpreted as the probability to request supervision at each iteration. In the `random` setting, in the first iterations the vast majority of the videos contains brand new objects, while in the last iterations the probability of encountering something new is close to zero. The `devel` setting spreads the encounter of objects evenly, except for the last iterations when almost all objects have already been seen. This latter setting is highly beneficial in terms of user effort, as the amount of supervision that FOLLOWER requests to match the performances of FULLOWER is far less than the one it needs in the random case, in agreement with what believed about infant learning [12].

The middle row in Figure 3.3 shows the "instantaneous" accuracy in classifying the next object (as brand new or as one already seen). This can be computed as:

$$\mathbb{1}\left(\begin{array}{c} \delta(v_s, v_{min}) \leq \lambda_r \wedge \texttt{same}(v_s, v_{min})) \\ \vee \\ \delta(v_s, v_{min}) > \lambda_r \wedge \nexists\, r'_s \in \mathcal{M} : \texttt{same}(v_s, r'_s) \end{array}\right) \tag{3.4}$$

where λ_r is the recognition threshold in eq. 3.3 (asking to the user is not an option here), $\mathcal{M}$ is the memory of the algorithm (at each training iteration), v_{min} is the nearest neighbor in $\mathcal{M}$, and `same` is true if v_s and v_{min} are representations of the same object. For both settings, the drop in performance follows the shape of the probability of encountering new objects (that is, it decreases as the number of objects to recognize increase, like in the re-identification experiments shown in Fig. 3.2). By design (i.e., choice of the α values) the

performance of the FOLLOWERs never fall much below the ones of the FULLOWERs (the relative difference in performance in the last 100 iterations is lower than 5%). At the start of the experiment, the values of the instantaneous accuracy of FULLOWER and FOLLOWER are roughly the same in the `random` setting, while in the `devel` setting FULLOWER has a small advantage over FOLLOWER. This is due to the considerably lower amount of supervision that FOLLOWER receives in the `devel` setting with respect to FULLOWER. This has the effect of momentarily reducing the performance of FOLLOWER in the first iterations (in which the algorithm is learning thresholds λ_l and λ_u). After this initial phase, the gap between the two algorithms shrinks. In both settings the performance increase towards the end, due to the fact that the algorithms are exposed mainly to objects they have already seen.

We also evaluated the performance of the algorithms on a separate *in-training* evaluation set, as customary in online learning settings. The lower row in Figure 3.3 shows the fraction of correctly recognized objects (in red for FULLOWER, in blue for FOLLOWER):

$$\frac{1}{|S_{T_e}|} \sum_{s \in S_{T_e}} \mathbb{1}\left(\delta(v_s, v_{min}) \leq \lambda_r \wedge \texttt{same}(v_s, v_{min})\right)$$

and the fraction of examples correctly identified as unseen objects (in black for FULLOWER, in purple for FOLLOWER):

$$\frac{1}{|S_{T_e}|} \sum_{s \in S_{T_e}} \mathbb{1}\left(\delta(v_s, v_{min}) > \lambda_r \wedge \nexists\, r'_s \in \mathcal{M} : \texttt{same}(v_s, r'_s)\right)$$

over the 90 hold-out videos S_{T_e} (one per object). In the `random` setting, due to the fact that in the first iterations the overwhelming majority of the encountered object were never seen before, the two algorithms are strongly biased towards marking every object as unseen. For this reason the green and purple curves are overlapping.

In this setting, the advantage of receiving feedback at each iteration is apparent in the progressively larger gap in recognition accuracy. On the other hand, in the `devel` scenario the algorithms are exposed to both new and unseen object right from the beginning. As FOLLOWER requests supervision only for ambiguous cases (i.e., nearest-neighbor distance neither low nor high enough), it is more biased towards predicting objects as seen with respect to FULLOWER. This enables FOLLOWER to match FULLOWER performances in terms of recognition accuracy even when the number of novel objects shrinks towards zero.

The same trends were found when repeating these experiments over the CoRe50 dataset [53]. Figure 3.4 presents the results obtained over the `random` and `devel` setting, obtained using a value of α of 0.84 and 0.6 respectively. These two values were selected in order to minimize the amount of supervision in the two setting while keeping the performances at a level comparable with FULLOWER.

The results are in line with the ones obtained over our dataset. A `devel` setting still requires less effort from the user to enable FOLLOWER to match the performances of FULLOWER. Due to the fact that CoRe50 contains many sequences but fewer different objects (compared to our dataset), in the 90% of the iterations the models receive a sequence containing an already seen object. For this reason the FOLLOWER models make less queries on average for each iteration in the experiments involving this dataset.

The different number of sequences per object in the CoRe50 dataset, with respect to our dataset, led to have a distribution of unseen objects (the green curves in the upper row of graphs) that is quite similar between `random` and `devel`. In both settings almost all the objects are shown to the models in the first 200 iterations. This is due to our choice to keep the parameters of the `devel` setting, i.e. the probability of encountering a new object, at the same value of the ones used in the experiments showed in the Figure 3.3. As a result, the graphs of the instantaneous accuracies and the recognition performances of seen and unseen objects, comparing the two settings, have a similar shape.

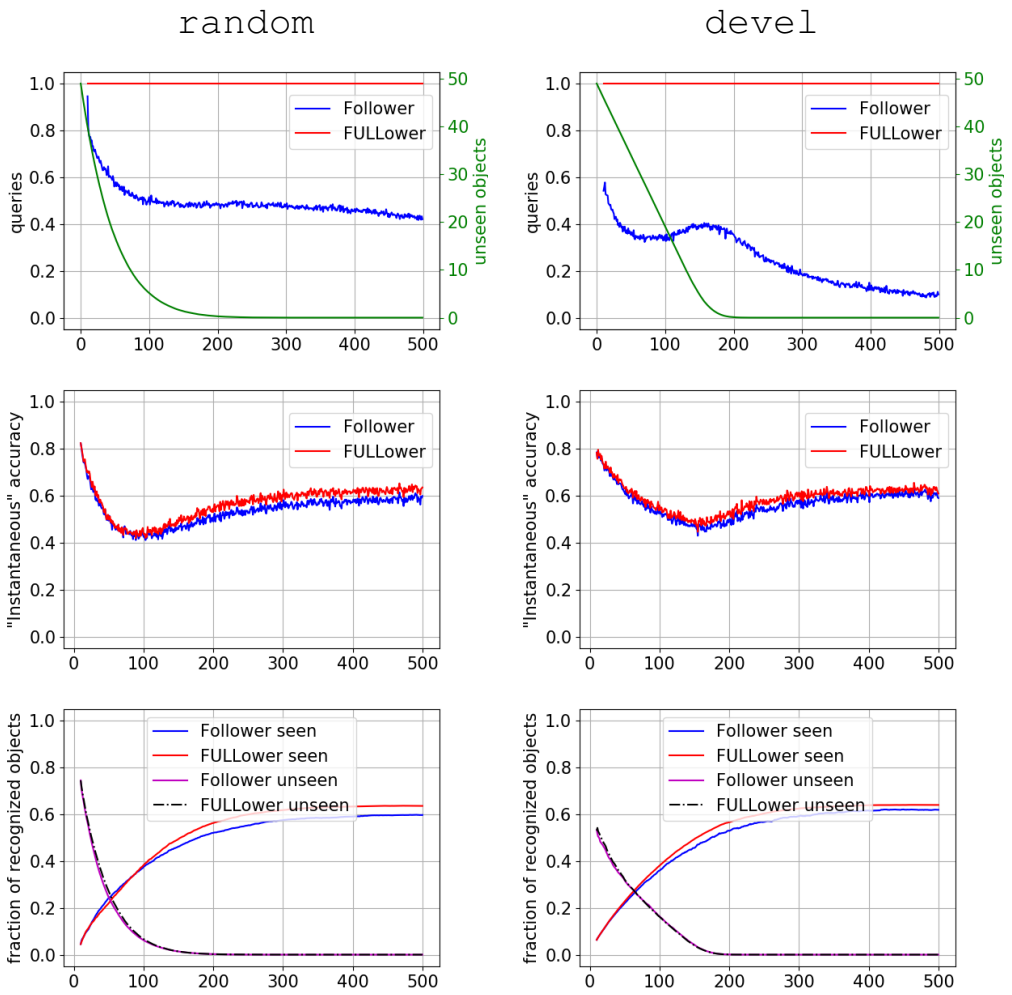


FIGURE 3.4: Open world recognition results on the CoRe50 dataset for increasing number of iterations (on the horizontal axis).

Unsupervised

The last series of tests are aimed at measuring the ability of the model to autonomously recognize and discriminate new objects. These tests are performed *after* the interactive training session with the user. The model is presented with all the 50 sequences of the 10 objects that were kept separate and never shown. The role of the model is to classify and store them as in the interactive phase, but without resorting on the supervision of the user. As for the training phase, we tested the performance of our algorithm by showing these 50

sequences both with a `random` and a `devel` policy[2].

In this phase we computed the overall accuracy of the model, averaging the instantaneous accuracy (eq. 3.4) over all 50 decisions, one for each sequence. We refer to this metric as the Averaged Instantaneous Accuracy (AIA). As in this phase no supervision is provided by the user, this evaluation can be seen as a sort of incremental clustering. At each iteration the model observes a new sequence and decides whether to consider the object in it as the same as one seen previously (linking it to its nearest-neighbor in memory), or to add it as a novel object. Let's consider a graph with sequences as nodes and an edge between each pair of nodes predicted as representing the same object. The connected components of this graph can be seen as clusters, each representing a (presumably) distinct object. We extracted these clusters for the 50 evaluation sequences and we computed two widely used clustering metrics, the Adjusted Mutual Information score (AMI) and Adjusted Rand Index score (ARI).

Table 3.1(top) shows the results obtained when presenting evaluation sequences using a `random` policy. The table shows the metrics described above, computed over FOLLOWER trained using a `random` (left) or a `devel` (right) policy. In order to compare the two training policies over the same number of examples, we tried different values of α so as to end up with an approximately equal number of training examples ($|\mathcal{K}|$ in the table), for a decreasing number of training examples. It is easy to see that when trained with the `random` policy, the performance of FOLLOWER are seriously affected by a decrease in the number of training instances. On the other hand, a `devel` policy during training enables the model to retain similar levels of accuracy even for substantially reduced amount of user feedback. While the clustering performance are more affected by the reduced level of supervision, this effect is much less pronounced when using a `devel` policy during training with respect to a `random` one.

[2]Given the low number of different objects, we decided not to compute the unsupervised evaluation on the CoRe50 dataset, in order to avoid to further reduce the number of objects in supervised evaluation

TABLE 3.1: Test phase evaluation results. Top results refer to a `random` evaluation policy, bottom ones to a `devel` one. Results are computed over FOLLOWER trained using a `random` (left) or a `devel` (right) policy.

		`random`				`devel`		
	$\|\mathcal{K}\|$	AIA	ARI	AMI	$\|\mathcal{K}\|$	AIA	ARI	AMI
`random`	285	0.47	0.45	0.38	289	0.46	0.44	0.38
	237	0.45	0.43	0.37	240	0.46	0.42	0.36
	210	0.43	0.39	0.34	203	0.45	0.42	0.36
	177	0.4	0.35	0.31	152	0.45	0.41	0.35
	81	0.32	0.21	0.2	84	0.45	0.4	0.38
	36	0.33	0.22	0.21	37	0.45	0.35	0.37
`devel`	285	0.73	0.56	0.5	289	0.73	0.6	0.54
	237	0.73	0.49	0.45	244	0.73	0.56	0.52
	210	0.72	0.44	0.4	203	0.73	0.55	0.5
	177	0.71	0.37	0.35	152	0.73	0.52	0.49
	81	0.68	0.21	0.2	84	0.73	0.44	0.45
	36	0.69	0.22	0.21	37	0.72	0.35	0.4

Table 3.1(bottom) shows the results obtained when presenting evaluation sequences using a `devel` policy. The trends seen with the `random` evaluation policy are basically preserved here, with a `devel` policy at training time leading to better results than a `random` one. Comparing the two tables, it is apparent that FOLLOWER achieves massive improvements in recognition accuracy when evaluated with sequences in a `devel` rather than `random` order, with up to 100% increase in AIA. Clustering performance are also substantially better in most cases, especially when the models is trained with a `devel` policy. Note that the decrease in clustering performance when reducing the number of training examples is more pronounced here with respect to the top part of the table. This is due to the fact that a `devel` evaluation policy allows the method to substantially increase its ability to recognize already seen objects, at the cost of a relative decrease in ability to identify the unknown. In absolute terms, however, also in terms of clustering the best results are achieved when FOLLOWER is both trained and evaluated with a `devel` policy.

3.3 Summary

In this Chapter we presented the first results of a long term project whose goal is to build systems which see, perceive and interact in open world environments like humans. Our results show that FOLLOWER is capable of progressively memorizing novel objects, even in complete absence of supervision, and that a developmental strategy is highly beneficial in boosting its performance and reducing its need for human supervision.

In this line of thinking the natural evolution of this work is to make the algorithms capable of dealing with semantics and make it knowledge-aware, this being the basis for a meaningful interaction with humans. The key choices underlying this work (i.e., the exploitation of persistency, the choice of a similarity-based approach and the attention to incrementality) were indeed motivated by this intuition. The use of similarity and the focus on the instance-level allow for the adaptability to an Open World environment, with instance-level recognition naturally scaled to the perception of classes. This is the central focus of the work that will be presented in the next chapter.

Chapter 4

A new hierarchical theory

The Oxford Research Encyclopedia defines *Lexical Semantics* as the study of *word meanings*, where these word meanings are implicitly assumed to be those constructed by humans when hearing a word. Following this definition, we take the meaning of a word to be the the mental representation which is constructed when hearing or reading a word, i.e., its *concept* [49]. Very much in the same line of thinking, in this Chapter we on *Visual Semantics*, which we take as the study of how to build concepts when using vision to perceive the environment. We tackle the problem with a computational approach, motivated by the goal of building a machines which rely on meaningful human machine interactions [26].

The key observation underlying our work is that the concepts built via language, e.g., the concept associated to the word *cat*, are built according to a process which is very different from that used when perceiving something, e.g., *a cat*, via vision; it is a fact that in humans the two processes involve different parts of the brain [56]. Following the terminology defined in [29], we call the concepts built from words, *classification concepts*, and the latter concepts, *substance concepts*. This terminology is motivated by the fundamentally different *function* that these concepts have, the first being in charge of *recognizing substances* as they are perceived, the latter being in charge of *classifying what is being perceived* according to various organization principles. This idea of seeing concepts as (biological) *functions*

is based on the work in the field of *Teleosemantics*, sometimes called *Biosemantics* [55], and in particular on the work by the philosopher R. Millikan [64, 61, 65, 63]; some of the implications on AI of the distinction between substance concepts and classification can be found in [29, 22].

Despite the crucial differences existing between substance and classification concepts, humans are able to seamlessy switch between them, being able to always use suitable words to describe what they perceive; this despite (and maybe, thanks to) the many possible combination of words that language provides them and the many different situations in which the same reality presents itself to them. Our goal is to provide a computational model that can cope with this phenomenon, namely of the process by which *a substance, e.g., a cat, as it appears in one or more sequence of images, becomes represented by a substance concept which is in one-to-one correspondence to the classification concept of the word naming it, e.g., the word cat*. A more precise articulation of the problem we deal with is as follows.

Suppose that a person and a machine, e.g., a pair of smart glasses, are put in a situation where they both see exactly the same parts of the world under the same visual conditions. This interaction between the human and the machine is supposed to last for very long, possibly for as long as the life of the person, or of the machine. Suppose that the human is an adult, namely a person with a full repertoire of words and concepts which allow her to describe what she sees according to her current point of view, which will change in time as a function of many internal and external factors, not least depending on how objects appear to her.[1] *Dually, assume that the machine starts from scratch without any prior knowledge of the world and of how to name whatever it perceives. How can we build an algorithm which, by suitably asking the human, will learn how to recognize and name whatever it sees in the same way as its friend of a life?*

[1] We assume that this person does not make mistakes and that she is consistent in the ways she names what she sees.

Put it in simple words, the problem that we are dealing with is how to develop a learning algorithm which will be able to recognize the outside world, evolving in time and adapting coherently with the needs of the user. A meaningful metaphor for this problem is that of a mother who is teaching her baby child how to name things using her own words in her own spoken language. The work in [29] provides an extensive description of the challenges related to this problem, mainly related to the many-to-many relation existing between what is seen and what is conceptualized. Further complications come from the fact that, based on the definition provided above, the learning algorithm needs to satisfy the following further requirements:

- it must be generic, in that it should make no assumptions about the input objects;
- it must learn new objects never seen before as well as novel features, never seen before, of previously seen objects;
- it must learn from a small number of examples, starting from no examples.

We propose a general Knowledge Representation (Knowledge Representation) *theory* which addresses the problem above. This theory is articulated in terms of a set of novel definitions of some basic notions, most importantly that of *object*. The theory proceeds as follows.

- We model *objects* as *substance concepts*, which in turn are modeled as sets of *visual objects*, i.e., sequences of similar frames, as perceived in multiple events called *encounters*. Visual objects are stored in a *cumulative memory* $\mathcal{M}$ of all the times they were previously perceived.
- Substance concepts are organized into a *(visual) subsumption hierarchy* which is learned based on two notions, those of `Genus` and `Differentia`, which replicate the notions with the same name that, in *Lexical Semantics*, are used to build

subsumption hierarchies of word meanings [60, 28]. The key observation is that the one-to-one correspondence which must exist between two hierarchies in order to be able to speak meaningfully, i.e., in human terms, of what is perceived is not a given nor something easy to compute.

- The visual hierarchy is learned autonomously by the algorithm; the user feedback makes sure that the hierarchy built by the machine matches her own linguistic organization of objects. In other words, the user feedback is the means by which the hierarchy of substance concepts is transformed into a hierarchy of classification concepts. The key observation here is that the user feedback is provided not in terms of object names, as it is usually the case, but in terms of the two properties of `Genus` and `Differentia`.

The chapter is organized as follows. First, we introduce objects as classification concepts, as they are used in natural language and organized in Lexical Semantics hierarchies (Section 4.1). This section provides also an analysis of why the very definition of classification concepts makes them unsuitable for visual object recognition. Then we define substance concepts as sets of visual objects, where a visual object is taken to be a sequence of frames as they occur, for instance, in a video (Section 4.2). Then, in Section 4.3, we provide the main algorithm by which substance concepts are built, while, in Section 4.4, we describe how a hierarchy of substance concepts is built which is aligned with that of classification concepts. In this section we also provide the two basic notions of `Genus` and `Differentia` which are used to build the hierarchy. The setting and the algorithm for object learning is described in Section 4.5. This algorithm has been developed for the base case of hierarchies of depth two. The extension to hierarchies of any level is discussed in the next chapter. The algorithm in evaluated in Section 4.6. Finally, the chapter ends with a summary of the work presented (Section 4.7).

4.1 Objects as Classification Concepts

Let us concentrate on nouns, as they are usually used to name objects. In Lexical Semantics the meaning of nouns is provided via definitions articulated in terms of `Genus` and `Differentia` [60]. Let us consider for instance the following two definitions:

- a *triangle* is a *plane figure* with *three straight bounding sides*;
- a *quadrilateral* is a *plane figure* with *four straight bounding sides*.

In these two definitions we can identify three main components:

- `Genus`: some previously defined set of properties which is shared across distinct objects, in the definitions above the property of *being a plane figure*;
- `genusObj` (also called `genusObj` object): a certain representative object which satisfies the `Genus` property, in the definition above, the object *plane figure*. The set of objects satisfying the *Genus* properties are said to have that (same) `genusObj`;
- `Differentia`: A selected novel set of properties, different from the `Genus` properties, which are used to differentiate among objects with the same `genusObj`, in the definitions above, the properties *having three straight bounding sides* and *having fours straight bounding sides*, defining, respectively, triangles and quadrilaterals as distinct objects with the same `genusObj`.

`Genus` and `Differentia` satisfy the following four constraints:

- *Role 1 of* `Genus`: if two objects have different `genusObj`, then they are (said to be) *different*. For instance, a pyramid is not a plane figure and, therefore, is different from a triangle.

- *Role 2 of* `Genus`: The inverse of Role 1 is not true, namely we may have different objects with the same `genusObj`. For instance, a quadrilateral and a triangle are both plane figures but they are not the same object.

- *Role 1 of* `Differentia`: Two objects with the same `genusObj`, but different from the `genusObj`, are (said to be) the *same* object if and only if the `Differentia` properties do not hold of the two objects. Thus, for instance, two objects with the same `genusObj` and with a different `Differentia`, e.g., a triangle and a quadrilateral, are different despite being both a plane figure. Dually, two objects with the same `genusObj` and the same `Differentia`, e.g., two triangles, are the same object (relatively to the current selection of `Genus` and `Differentia`).

- *Role 2 of* `Differentia`: a `genusObj` and an object with that `genusObj` are different when the latter is characterized by a set of properties, i.e., its `Differentia`, that the `genusObj` does not have. Thus for instance a triangle is not the same as a plane figure, as it is just one of the many possible plane figures", e.g., triangles, quadrilaterals which share the same `genusObj` (in this case, plane figures).

The first observation about the definitions above is *what it means for two objects to be the same object*. It definitely does not mean that they are the same instance, it only means that the two objects satisfy the same `Genus` and the same `Differentia`. Thus for instance a right triangle and a equilateral triangle are the same object, when compared with quadrilaterals, in that they have the same number of sides, but they are different when the `Differentia` being considered is the triangle internal angles. This observation has two immediate consequences. The first is that the process above can be iterated at any level of detail, thus creating hierarchies of any level of depth. It is a fact that, in lexical semantics, the meaning of nouns is organized as a hierarchy of increasing specificity, each layer being characterized by a new `Genus` and a new `Differentia`, where an object with a certain

`Genus` is a child of its `genusObj`. Such a hierarchy of depth n can be seen as the recursive stacking of $(n-1)$ hierarchies of depth 2, where the `genusObj` of the depth 2 hierarchy one level down is one of the children of the `genusObj` one level above. The root of this hierarchy is usually referred as *thing* or entity [60, 28]. This process of progressive differentiation allows to split the set of objects under consideration into progressively smaller and smaller sets, based on the selected set of properties.

The second observation is that the above definitions state what is the same of what is different using natural language. These linguistic definitions are designed to generate what we call *classification concepts*, namely concepts which are amenable for classification [22]. And in fact the very existence of lexical semantics hierarchies provides evidence of their suitability for this task. This type of definitions is well grounded in the everyday practice, in particular when used to name and describe things, for instance during interactions among humans. However they do not work as well while one is in the middle of the recognition process, namely while she is trying to identify the object she is looking at. How many times were you able to recognize someone or something based only on a natural language description, without the help of a photo or anything which could point to specific spatial objects or properties?

Let us clarify the problem highlighted by the last observation with an example. Assume you see at a certain distance two *things* moving towards you. Initially you will not recognize what these things are but, when they are close enough, you will be able to recognize two *person*s, seen from the back. The day after, you see again two persons, which may or may not be those recognized the day before: hard to say, they did not come close enough. In any case, this second time these two persons get close enough for you to finally recognize your friends *Karl* and *Frank*. What allowed you to distinguish *Karl* from *Frank* is that the former has white hair while the latter has black air and mustaches. Later on, walking towards you, you will recognize a *woman*. You will have been able to recognize her as a

person different from the two previous *men* because she has long hair and a skirt. Of course you will know the terms you have used to describe what you will have seen, i.e., *person*, *man*, *woman*, *Karl* and *Frank*, as someone will have taught them to you, for instance during your childhood. In Knowledge Representation, the simple scene described above can be formalized by saying that *Karl* and *Frank* are *instances*, while *person*, *man* and *woman* are *(classification) concepts* and by stating the following facts: *man(Karl)* (to be read as *Karl is a man*), *man(Frank)*, *man* $\sqsubseteq$ *person* (to be read as *man is subsumed by person*) and *woman* $\sqsubseteq$ *person*, the latter two facts stating that all men and all women are persons. The resulting hierarchy, as formally defined via the logical subsumption symbol $\sqsubseteq$, is provided in Figure 4.1 (first left) where the classification concepts there represented are defined, for instance, as (partial quote from [60])

- *person*: individual, someone, somebody;
- *woman*: an adult female person;
- *man*: an adult male person;
- *Karl*: an instance of a man;
- *Franz*: an instance of a man.

Notice how the above definitions and the properties they involve (e.g., being adult, male or female, being an instance) are completely unrelated to the process by which recognition was carried out, which was in terms of a continual analysis of visual information, at increasing levels of precision.

The previous example is representative of the situation where the observer has complete knowledge of the objects being perceived and the partiality of information is caused by some contextual factors. Consider now the hierarchy of classification concepts in the center of Figure 4.1, which names and classifies daisies, whose images are in the corresponding

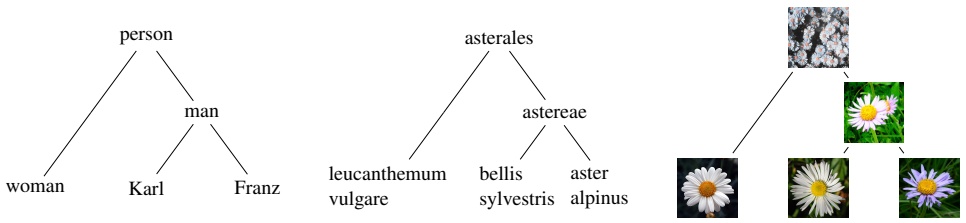


FIGURE 4.1: (Left): a classification concept hierarchy; (Center): a classification concept hierarchy for daisies; (Right:) the center hierarchy where words are substituted with images representing the corresponding daisies.

place in the hierarchy in the right of Figure 4.1. A possible lexical semantics definition of these daisies is as follows:

- *asterales*: an order of flowering plants containing eleven families, the most notable being asteraceae (known for composite flowers made of florets);
- *leucanthenum vulgare*: flower native to Europe and the temperate regions of Asia, commonly referred as marguerite;
- *astereae*: a tribe of plants, commonly found in temperate regions of the world, also called daisy or sunflower family;
- *bellis sylvestris*: Southern daisy, perennial plant native to central and northern Europe;
- *aster alpinus*: blue alpine daisy, plant commonly found in the mountains in Europe.

Most readers, in particular those who are not florists, even if coming to know about the hierarchy above, e.g., because being described it, will be unable to recognize the various types of daisy. As a consequence they will not be able to build it starting from images (e.g., the ones on the right in Figure 4.1), simply because they will not be able to recognize the features which allow to distinguish among the various types of daisy. Most likely, in many cases, the hierarchy will be collapsed to a single node while, in others, the light purple daisy will be separated from the others, just because of its colour.

In general, classification concepts do not seem well suited for the process of object recognition. This despite the fact that it is common practice to use them in supervised machine learning, where the user feedback is, often if not always, provided in terms of *words* whose meaning is defined via lexical semantics hierarchies. Evidence of this difficulty is provided by the so called *Semantic Gap problem*, which was already identified in 2010 [92] as (quote) "*... the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.*", and which is still unsolved. The main motivation for this fact seems to be that classification concepts model objects as *endurants*, i.e., as being always wholly present, at any given moment in time, with their proper parts being present in a certain spatial configuration and satisfying certain properties (e.g., color, shape, position, activity) [24]. Of course, this configuration may change, or the object might not be accessible visually (as in the first example above), or the observer might not be able to discriminate some of its relevant properties (as in the daisies example above) but this has no impact on how classification concepts are defined. Classification concepts, while well serving the purpose of describing what was previously perceived, are completed unrelated to the process by which the objects are perceived and, in particular, to the fact that their perception is constructed *incrementally*, via a set of *partial views* which progressively enrich what is visually known. To this extent, notice how *person*, *man*, and *Karl* are correctly represented in Figure 4.1 as three different classification concepts. However in the little story above, these three classification concepts actually describe the same piece of reality, seen at different times, at different levels of detail and from different points of view.

4.2 Objects as substance concepts

The key intuition underlying the work described here is to model objects are *perdurants*, where, quoting from [24] ... *perdurants ... just extend in time by accumulating different temporal parts, so that, at any time they are present, they are only partially present, in the sense that some of their proper temporal parts (e.g., their previous or future phases) may be not present.* According to these definitions, examples of endurants are physical objects, e.g., those mentioned above, while examples of perdurants are events and activities. Taking an object as a perdurant amounts to saying that we never have a full (visual) picture of the object but that the full representation is built progressively, in time, as it is *always* the case in our everyday life. We call *substance concepts* the representation of objects as perdurants.

The starting point in the definition of substance concepts is the crucial distinction between what we perceive as being in the real world, that we call *substances* and their corresponding mental representations, i.e., their substance concepts. Following R. Millikan, we take substances as those things (quote from [65]), "... *about which you can learn from one encounter something of what to expect on other encounters, where this is no accident but the result of a real connection ...*". [29] provides a detailed discussion of what substances are and of how they generate substance concepts in the mind of observers, based on the work on *Teleosemantics* [55], and in particular on the work by Ruth Millikan [64, 60, 62, 66, 61, 65]. In the following, substances should be intuitively thought as those things which, when perceived in the most detail, will generate the perception of individuals, e.g., *Karl*, *my cat*, but that, under different conditions, will generate more generic or even very different substance concepts, e.g., *a moving object*, *an animal*. The key observation is that, while substances are crucial in our informal understanding of perception in that they allow us to focus on the process of how objects are perceived, they play no role in the formal model that we define below. With this in mind in the following: (i) we avoid defining what a substance

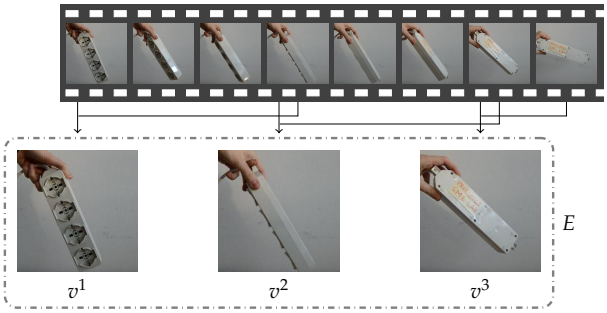


FIGURE 4.2: Example of an encounter. For better visualization, each visual object is represented, here and later, as its first frame.

is (no such definition could be meaningfully grounded in human experience); and (ii) we consider substances only in their *causal role* on the generation of a concept, a role that is constrained within the events during which a substance is perceived. We call such events, *encounters* and (iii) we qualify this causal role in terms of two properties that substances have, as introduced below, and that we call *Space Persistency* and *Time Persistency*. Notice however that both *Space Persistency* and *Time Persistency*, as all the definitions provided in this chapter, are given as properties of substance concepts.

We use superscripts to mean elements of a sequence, and (optionally) the subscript S, to mean elements obtained from one or more encounters E_S with the substance S, as in f_S^i, v_S^i, and O_S^i. Different subscripts mean elements generated in possibly different sets of encounters. In the following we omit the subscript whenever the substance we are referring to is clear from the context.

We assume that encounters are represented as *spatio-temporal worms*, i.e., temporal sequences of *frames*, where f_S^i is a frame for a substance S, each frame being encoded via a set of *low-level visual features*. We represent encounters, by exploiting the *Space Persistency* of substances, namely the fact that, in time, substances change very slowly their spatial position. Because of space persistency, during an encounter, any two adjacent frames will be very similar, while this will not necessarily be the case with two non adjacent frames. We model Space Persistency in terms of *Frame Similarity (Dissimilarity)*, written

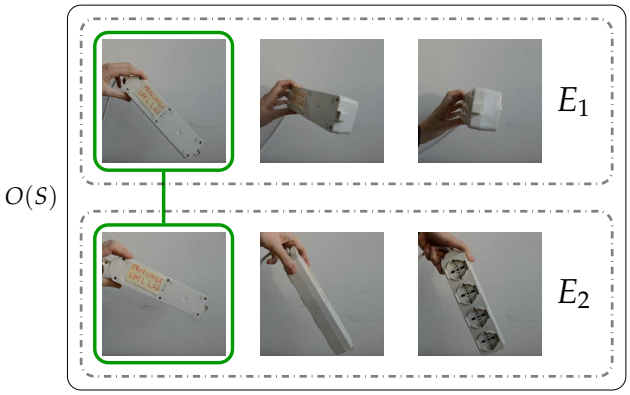


FIGURE 4.3: A single object consisting of two encounters. The green line connects two similar visual objects.

$f_{S_1} \simeq f_{S_2}$ ($f_{S_1} \not\simeq f_{S_2}$) (see the algorithm below for a possible implementation). Given Frame Similarity, we define *visual objects*, where v_S is a visual object for a substance S, as sequences of adjacent frames where the last frame is *similar* to the first, and *encounters* E_S as sets of visual objects, i.e.,:

$$E_S = \{v_S^1, \ldots, v_S^n\}. \tag{4.1}$$

Figure 4.2 reports an example of an encounter consisting of eight frames organized in three visual objects. Notice how having multiple similar frames in the same visual object makes it quite robust to local contextual variations. The first time a substance S is perceived as a new *object*, that object will consist of a single encounter; but this object will be enriched by subsequent encounters. We model this situation by taking objects to be the set of all the different visual objects collected by the different encounters. Let $E_S^1, ..., E_S^m$ be a set of encounters. Then we have:

$$O_S = \cup_i E_S^i = \{v_S^1, \ldots, v_S^n\} = \{v_S^i\}. \tag{4.2}$$

This situation is well represented in Figure 4.3 where each row is a different encounter.

4.3 Building Substance concepts

Objects as substance concepts get cumulatively built in time. Let $E_S^1, ..., E_S^m$ be a sequence of encounters. Let O_S be an object defined as in Equation (4.2). Then O_S is incrementally constructed as follows:

$$
\begin{aligned}
&1.\ \textsc{AddObject}(\mathcal{M}, E_S^1) && (4.3)\\
&2.\ \textsc{updateObject}(\mathcal{M}, O_S^i, O_S^{i-1} \cup E_S^i), && i = 2, ...\\
&3.\ O_S \textsc{ is } O_S^i, && i = 1, ...
\end{aligned}
$$

where:

- AddObject creates a new object O_S^1 in the *cumulative memory* $\mathcal{M}$ of the objects perceived so far;
- O_S^i is an object as perceived after any given number i of encounters; and
- $O_S^{i-1} \cup E_S^i \in \mathcal{M}$ is the cumulative memory of O_S^i;
- updateObject updates the current memory O_S^{i-1} of an object with the visual objects coming from E_S^i, thus constructing O_S^i;
- The construct is in Item 3 is the formal statement assessing that we take objects as the cumulative memory of what has been perceived so far.

A first observation is that item 3 implicitly states that substance concepts evolve in time, i.e., that they are perdurants. In this perspective, O_S^{i-1}, O_S^i, E_S^i and also O_S, are all partial views of S that individually contribute to the construction of O_S. This process of object construction will eventually terminate if the appearance of an object does not change, in that new encounters will not enlarge the set of visual objects defining the object. However an object can also keep evolving. Thus, for instance, the current encounter with *Frank* may

contain visual objects which are quite dissimilar from the ones encountered earlier on, for instance because we saw him at different ages (e.g., when he was fifteen and now that he is thirty-five).

A second observation is that the process described in Equation (4.3), and in particular the decision of which between step 1 and step 2 must be applied, depends on the ability to recognize whether the current encounter is a partial view of an object already recognized. But how to decide? Let us write $O_{S_1} = O_{S_2}$ to mean *Object Identity*, namely that the two substance concepts are actually two (partial) views of the *same* object, rather than two views of two *different* objects. Notice that this can be the case also in the case of objects generated by two different sequences of encounters and despite being, at least partially, visually dissimilar. Let us also write $O_{S_1} \neq O_{S_2}$ to mean *Object Diversity*. Then, Item 2 will be applied only for that object O_S such that $O_S^{i-1} = E_S^i$, while Item 1 will be applied whenever $O_S^{i-1} \neq E_S^i$ for all objects in $\mathcal{M}$.

The complications arising in the decision on Object Identity depend on two main factors. The first is that the *correlation between substances and substance concepts is many-to-many*. To reiterate an example from the previous section, the same substance can be perceived as *Karl*, as a *man* or as a *person* while, vice versa, the same substance concept, e.g., *man* can be recognized from multiple individuals. In other words, we need to decide at which level, in the visual subsumption hierarchy, the current encounter for the same substance should be assigned. The second issue is that, independently of the level of the subsumption hierarchy, the decision on Object Identity must made taking objects to be endurants, as represented by classification concepts, being classification concepts what is used by humans in their everyday interaction and classification activities. Object Identity is a much richer notion than visual similarity as it involves considerations like language, culture, function of the objects, and much more, see, e.g., [35, 70, 31]. Among other things notice how we have $O_S = O_S^i,\ i = 1, ...$, this meaning that Object Identity is invariant

in time. As a consequence, there is a *many-to-many correspondence between substance concepts and classification concepts*, as also extensively exemplified in [29].

The double many-to-many mapping from substances to classification concepts is the main cause of the inherent arbitrariness which exists in how to identify what we perceive as objects. This phenomenon is well known in Computational Linguistics and it is the cause of the so-called *lexical gaps*, namely concepts which are lexicalized in one language but not in other languages [27]. Things are made even worse when, even within the same language, one considers the subjective behaviour of individuals; see the two examples in Section 4.1. Notice that the problem is not that of constructing a hierarchy of meanings; in Section 4.4 we will show how this can be done based on the visual similarity of objects as defined as in Equation (4.2). The problem is that such a hierarchy will almost inevitably suffer from the Semantic Gap problem and, therefore, will not achieve the goal of aligning classification concepts and substance concepts. The solution we propose is articulated in the following main assumptions:

1. We assume that the fact that two objects are visually similar is a *necessary condition* for object identity. This assumption is well grounded in our everyday experience and also made in the mainstream Computer Vision research. To this extent, we introduce the notions of *Visual Object Similarity (Dissimilarity)*, written $v_{S_1} \simeq v_{S_2}$ ($v_{S_1} \not\simeq v_{S_2}$) and of *Object Similarity (Dissimilarity)*, written $O_{S_1} \simeq O_{S_2}$ ($O_{S_1} \not\simeq O_{S_2}$). Notice that we need to define what visual similarity is, given that, as discussed above, the same object can appear in many different ways; this will be discussed in Section 4.5.2.

2. We assume, as also implicit in Millikan's quote, that substances have a property of *Time Persistency*, namely some form of time invariance in how they appear across encounters. This assumption allows us to compare, up to a point, visual objects coming

from different encounters. Notice that how space and time persistency operate is specific to the objects being considered, no matter whether instances or concepts. Thus, for instance, *Karl* will keep having white hair while *Frank* will keep having black hair and mustaches, while, for instance, humans, like all animal species, are characterized by a *homeostatic mechanism* which causes them to possess a certain set of common traits (e.g., their shape, how they move) that often, but not always, make them look similar [29]. The key consideration here is that, once an observer has subjectively decided what is the object that she is trying to recognize from a substance S, the criteria for object identity do not change. In other words, time persistency applies not only to the perceived object but also to the perceiving subject.

3. We organize objects in a visual subsumption hierarchy, exactly like the one used in lexical semantics, but with the key difference that `Genus` and `Differentia` are computed in terms of the substance concepts' visual properties, as represented by the visual objects. This allows to deal with the problem of the many-to-many mapping between substances and substance concepts.

4. *Last, but not least*, we deal with the many-to-many mapping between substance concepts and classification concepts by relying on the key role of the *user supervision*. This transformation is crucial to the integration of visual perception, where objects keep evolving in time (i.e., they are perdurants), and language-based reasoning, which thinks of objects as being completely described in any moment in time (i.e., they are endurants). `Genus` and `Differentia` can be computed in a completely unsupervised manner, via object similarity, but the user feedback, which is given *only* on `Genus` and `Differentia`, guarantees that the machine-built hierarchies largely coincide, modulo recognition mistakes, with the hierarchies that a user would build.

Notice how this supervision is unavoidable, that it is exactly the same type of supervision that a mother would give to her child, and that it is subjective, evidence being also that different languages conceptualize different objects [27].

As a last remark, notice that in the visual hierarchy mentioned in item 3, all nodes are labeled only by substance concepts. Instead, in lexical semantics hierarchies, nodes are labeled by (classification) concepts, e.g., *man*, and instances e.g., *Frank*. In other words, as correctly pointed out by R. Millikan [65], but see also [29], from a perception point of view, the usual Knowledge Representation distinction between concepts (usually modelled as sets of instances) and instances does not apply.

4.4 Object Subsumption and Identity

As from Equation (4.2), objects, represented as substance concepts, are sets of visual objects. The idea is to exploit this fact to build a hierarchy of objects based on visual similarity. As from the discussion at the end of the previous section, this hierarchy gives us only the necessary conditions for object identity (*i.e.* visual similarity). In the following we will assume that the two hierarchies of substance concepts and classification concepts will coincide assuming, as also discussed at the end of the previous section, that the user feedback will be used to validate the choices made. The algorithm in Section 4.5.2 will show how this is done in practice by suitably asking feedback to the user.

As from the discussion in Section 4.1, a lexical semantics hierarchy can be seen as the iteration of many depth 2 hierarchies, each with its own `Genus` and `Differentia`. Therefore, without lack of generality, in the following we focus on hierarchies of depth 2. The main goal below is to restate the conditions for `Genus` and `Differentia`, informally stated in Section 4.1 for classification concepts, in terms of formally defined conditions on substance concepts. Let us assume that we are given a genus object `genusObj`. In the

general case the construction of `genusObj` will happen recursively from the top node, i.e., *thing* or *entity*. Then, let us define $\texttt{same}(O_{S_1}, O_{S_2})$ and $\texttt{Different}(O_{S_1}, O_{S_2})$ as two binary boolean functions which, based on the intuitions described above, discriminate over objects, defined as in Equation (4.2), based on their visual objects. We enforce the four roles in Section 4.1 by enforcing the following three constraints:

$$\neg\texttt{same}(O_{S_1}, O_{S_2}) \longrightarrow O_{S_1} \neq O_{S_2}, \tag{4.4}$$

$$\begin{gathered}\texttt{same}(O_{S_1}, O_{S_2}) \longrightarrow \\ (O_{S_1} = O_{S_2} \longleftrightarrow \neg\texttt{Different}(O_{S_1}, O_{S_2})),\end{gathered} \tag{4.5}$$

$$O_{S_G} \subseteq O_{S_1} \cap O_{S_2}. \tag{4.6}$$

where O_{S_G} is the `genusObj` of O_{S_1}, O_{S_2}. Notice how the specifics of `Genus` and `Differentia` are left open, we only require that they are both computed out of the visual objects in input, i.e., O_{S_1}, O_{S_2} and that they satisfy the three constraints above. This is on purpose as it gives us freedom in many dimensions, e.g., of the specifics of the learning algorithms used, of how visual similarity and/or object identity are defined, and also of how `same` and `Different` are defined in any different layer of the hierarchy under construction. The algorithm in Section 4.5.2 will instantiate the missing information selecting, for each decision point, one among the many possible options.

Let us concentrate on the constraints. They satisfy the following intuitions. *First*, they satisfy the four criteria defined in Section 4.1. Equation (4.4) formalizes *Role 1* and *Role 2* of `Genus` while Equation (4.5) formalizes *Role 1* of `Differentia`. Equation (4.6) formalizes *Role 2* of `Differentia`; in fact from Equation (4.6) we have $O_{S_G} \subseteq O_{S_1}$. To have $O_{S_G} \neq O_{S_1}$, O_{S_1} must have at least a visual object $v^i_{S_1} \notin O_{S_G}$. Then there are

two cases, either v_S^i is such that `Different` holds, in which case we are done (from Equation (4.5)), or this is not the case, in which case $O_{S_G} = O_{S_1}$, namely that visual object is irrelevant to the identity of O_{S_1}. Notice how this latter case does not rise if we take `genusObj` to be exactly the intersection. Equation (4.6) captures the intuition that the visual objects which are not considered belong to both objects by chance. Thus for instance, *Karl* and *Frank* might happen to have had, when observed, a red sweater. But red sweaters are not a characteristic of *men*. *Second*, Equation (4.4) captures the fact that `same` provides necessary but not sufficient conditions for object identity. *Third*, Equation (4.5) provides necessary and sufficient conditions for two objects to be different, but under the assumption that `same` holds. Namely, `Different` can be applied only after having discarded all the objects which do no satisfy `same`.

The three constraints above allow us to build the desired subsumption hierarchy. Let us write $O_{S_j} \sqsubseteq O_{S_i}$ ($O_{S_i} \sqsupseteq O_{S_j}$) and say that O_{S_j} *is subsumed by* O_{S_i} (O_{S_i} *subsumes* O_{S_j}) to mean that the visual objects of O_{S_j} are a subset of those visual objects of O_{S_i} which are relevant for the computation of `Different` (see discussion above on Equation (4.6)). We also write $O_{S_j} \sqsubset O_{S_i}$ and talk of *strict* subset and subsumption to mean $O_{S_j} \sqsubseteq O_{S_i}$ and $O_{S_j} \neq O_{S_i}$, and similarly for $O_{S_i} \sqsupset O_{S_j}$. Let us assume that O_{S_2} and O_{S_2} have the same `genusObj`, O_{S_G} namely, that $\texttt{same}(O_{S_1}, O_{S_2})$ and, therefore, $\texttt{same}(O_{S_G}, O_{S_2})$, $\texttt{same}(O_{S_G}, O_{S_1})$ hold. Clearly, $O_{S_G} \sqsubseteq O_{S_1}$ and $O_{S_G} \sqsubseteq O_{S_2}$. This makes the premise and therefore the consequence of Equation (4.5) hold of all three objects. We have the following cases (for compactness, below we write `D` to mean `Different`):

1. $\texttt{D}(O_{S_1}, O_{S_2})$, $\texttt{D}(O_{S_1}, O_{S_G})$ and $\texttt{D}(O_{S_2}, O_{S_G})$: we have $O_{S_G} \sqsubset O_{S_1}$ and $O_{S_G} \sqsubset O_{S_2}$, namely the situation where all three objects are different;

2. $\texttt{D}(O_{S_1}, O_{S_2})$, $\neg\texttt{D}(O_{S_1}, O_{S_G})$ and $\texttt{D}(O_{S_2}, O_{S_G})$: we have $O_{S_G} = O_{S_1}$ and $O_{S_G} \sqsubset O_{S_2}$;

3. $\texttt{D}(O_{S_1}, O_{S_2})$, $\texttt{D}(O_{S_1}, O_{S_G})$ and $\neg\texttt{D}(O_{S_2}, O_{S_G})$: we have $O_{S_G} = O_{S_2}$ and $O_{S_G} \sqsubset O_{S_1}$;

4. $\neg\texttt{D}(O_{S_1}, O_{S_2})$, $\texttt{D}(O_{S_1}, O_{S_G})$: we have $O_{S_G} \sqsubset O_{S_1}$ with $O_{S_1} = O_{S_2}$;

5. $\neg\texttt{D}(O_{S_1} O_{S_2})$, $\neg\texttt{D}(O_{S_1}, O_{S_G})$: we have $O_{S_1} = O_{S_2} = O_{S_G}$.

Two observations. The first is that, under the assumption that O_{S_1} and O_{S_2} have the same `genusObj` O_{S_G}, `same` and `Different` provide us with *necessary* and *sufficient* conditions for both *object identity* and *object subsumption* and, therefore, they provide us with the means for building the depth 2 sub-hierarchy under consideration. In fact as from the (only if) directions of clauses 2,3,4,5, two objects are the same if they have the same `Genus` and `Different` does not hold of them. Thus, taking into account the necessary conditions provided by Equation (4.4) we have:

$$O_{S_1} = O_{S_2} \quad \longleftrightarrow \quad \texttt{same}(O_{S_1}, O_{S_2}) \quad \wedge \quad \neg\texttt{Different}(O_{S_1}, O_{S_2}) \tag{4.7}$$

Furthermore, as from clauses 1,2,3,4, we have that `genusObj` is the parent node of the objects of which it is the `genusObj`, namely:

$$O_{S_1} \sqsubset O_{S_G} \quad \longleftrightarrow \quad \texttt{Different}(O_{S_1}, O_{S_G}) \tag{4.8}$$

The concluding remark is that, so far, we have only dealt with hierarchies of depth two, but the reasoning above can be replicated to build hierarchies of any depth. Let us assume that we have a new object O_{S_3} with $\neg\texttt{same}(O_{S_3}, O_{S_1})$ and, thus, $O_{S_1} \neq O_{S_3}$, $O_{S_2} \neq O_{S_3}$, and $O_{S_G} \neq O_{S_3}$. At the same time, O_{S_3} can share some visual objects with O_{S_1} or O_{S_2} which make `Different` false. Thus, for instance, a *plane* is not a *bird*, but they both *fly*. Given,

any two objects there is *always* a `genusObj`, also when these objects are very different, and this is the key fact which allows for the construction of subsumption hierarchies of any depth. Notice how we may end up with a `genusObj` which is the empty set, this being the limit case where the `genusObj` is *thing*: a generic object is something which has been perceived but with no associated visual objects.

4.5 The Framework

The theory described in the previous section has been implemented in an algorithm capable of recognizing hierarchies of depth two. In this section we present the setting in which this algorithm reasons, as well as its implementations details. An early release of the implementation of the entire framework is available[2].

4.5.1 The Setting

The setting for this framework can be considered as an extension of the one presented in Section 3.1.1. The main difference is the extension to the recognition of any kind of object, rather than single instances of objects. The limitations described in the setting of the previous chapter are still present, but new constraints negate the use of the same architecture.

For the same reasons explained in the previous chapter we decided to make use again on a pre-trained network. Taking advantage of recent developments in the field of Deep Clustering, we make use of a Neural Network trained with a self-supervised policy [10], thus avoiding the risk of poor intra-class representations.

Due to the requirements of the theory, a single representation (like the average of all the frames) is not suited for the task. This is due to the fact that a single representation would not allow for the clear distinction between the object and its `genusObj`. Thus from

[2]Early release: https://github.com/lucaerculiani/towards-visual-semantics

a sequence multiple representations were obtained by using a moving average run over the embeddings obtained from the single frames. The next section will explain in details how they were built and used.

The use of a running mean in place of an average requires some rethinking of the environment in which the input data is captured. As explained in Section 3.1.1, an average based aggregation trades robustness against noise with some loss of information. In order to retain the complete information one could use directly each single frame separately (as its own visual object). This obviously means increasing both the level of noise as well as the storage and computation requirements. Using a moving average is the middle ground between these two extremes.

In order to have this simple approach working, we found we had to reduce the amount of noise in the input data. This was done by creating a new version of the dataset using always the same neutral background (a white wall). This simplifying assumptions is clearly limiting for a real world application. This assumption is motivated by the focus on the recognition of `Genus` and `Differentia`, rather then on the distinction between objects and background. This setting requires a collections of objects that can be grouped on the basis of their visual appearance. Inside each group, all objects must have some partial views that make them *indistinguishable* from the other elements of the group (the `Genus`), while at the same time having other views that enable the discrimination of single objects (the `Differentia`). No public dataset enforces this constraint. As a consequence we have created our own data set, which consists of a collection of video sequences of various objects, recorded while rotating or being deformed against a blank background, making sure that each video contained only a *partial* view of the object. The resulting dataset contains five pairs of objects that only differ for a certain view (as in Figure 4.4). For each object in each pair, we recorded five videos that contain the discriminative view, and five that do not.

4.5.2 The Algorithm

We first provide a computational interpretation of the definitions introduced in the previous sections and then we introduce the algorithm, which should be seen as a first prototype and representative of a wide class of algorithms. Any algorithm will do as long as it satisfies the constraints for `Genus`, `Differentia` and the `genusObj`. This enables the development of smarter algorithms, for instance concentrating on noisy backgrounds or the many other complexities which make Computer Vision a very complex task. The algorithm we presents in this section is able to handle hierarchies of objects of two levels, and thus to encode just one single `genusObj` per object. We employed this strategy to keep the algorithm as simple as possible. This limitation will be eased in the next chapter.

Frames. We encode frames using an *unsupervised* deep neural network [10], trained to perform a combination of self-supervised and clustering tasks. Using an unsupervised network allows to produce embeddings which are not explicitly biased towards classes of objects, while, at the same time, complying to the assumption that machines extract features from what they perceive, autonomously, as humans do. We define *Frame similarity* as the Euclidean distance between frame encodings.

Visual objects. We define them as contiguous sequences of frames, and we represent them as the average between the frame encodings. We assume for robustness that visual objects are of a fixed limited length. Visual object are perceived by a procedure, named PERCEIVE, which returns an encounter as a set of visual objects, as from Equation (4.1). We model

Visual object similarity. as a diversity threshold on the distance between visual objects:

$$v^i \simeq v^j \stackrel{\text{def}}{=} d(v^i, v^j) < \theta \tag{4.9}$$

Objects. We define Objects as from Equation (4.2), i.e., as sets of visual objects extracted from sets of encounters. We define *object similarity*, analogously to visual object similarity,

as a diversity threshold on the distance between objects:

$$O_{S_1} \simeq O_{S_2} \stackrel{\text{def}}{=} d(O_{S_1}, O_{S_2}) < \theta \tag{4.10}$$

where the distance between objects is defined as the minimal distance between their respective visual objects:

$$d(O_{S_1}, O_{S_2}) = \min_{v^i \in O_{S_1}} \min_{v^j \in O_{S_2}} d(v^i, v^j) \tag{4.11}$$

By keeping the same threshold as for visual object similarity, we have that object similarity holds when two objects have at least two similar visual objects:

$$O_{S_1} \simeq O_{S_2} \iff \exists v^i \in O_{S_1}, \exists v^j \in O_{S_2} \ : \ v^i \simeq v^j. \tag{4.12}$$

`Genus`. We implement `Genus` as a Boolean function GEN which computes object similarity:

$$\text{GEN}(O_{S_1}, O_{S_2}) \stackrel{\text{def}}{=} O_{S_1} \simeq O_{S_2} \tag{4.13}$$

In other words, in this version of the algorithm we take object similarity to be a sufficient condition to GEN to hold.

`Differentia`. We implement `Differentia` as a boolean function DIF which holds for two objects with the same `genusObj` if there is no visual object, aside the ones in their `genusObj`, which make the two objects similar, namely:

$$\begin{aligned} \textsc{Dif}(O_{S_1}, O_{S_2}) \quad &\stackrel{\text{def}}{=} \quad \nexists v^i \in O_{S_1} \setminus \textsc{genOf}(O_{S_1}) \\ &\quad\;\; \nexists v^j \in O_{S_2} \setminus \textsc{genOf}(O_{S_2}) \\ &\quad\;\; v^i \simeq v^j \end{aligned} \tag{4.14}$$

where GENOF is a function which computes the `genusObj`, as:

$$\begin{aligned} \textsc{genOf}(O_{S_1}) \quad &\stackrel{\text{def}}{=} \quad \{v^i \in O_{S_1} | \exists O_{S_2} \in \mathcal{M}, \exists v^j \in O_{S_2} : \\ &\qquad O_{S_1} \neq O_{S_2} \wedge \textsc{sg}(O_{S_1}, O_{S_2}) \wedge v^i \simeq v^j\} \end{aligned} \tag{4.15}$$

The function $\textsc{sg}(O_{S_1}, O_{S_2})$ returns true if the user in the past gave supervision (see below), telling the algorithm that O_{S_1} and O_{S_2} share a `genusObj`. The way in which this function computes the `genusObj` is the reason that limits the applicability of this algorithm to hierarchies with just two levels. The unary function $\textsc{genOf}(O_{S_1})$ enables to separate the subset of the visual objects associated to the `genusObj` of O_{S_1} from all the others, based on the similarity of the visual objects of O_{S_1} with the visual objects of all the other objects seen so far. This obviously disallows to compute more than one `genusObj` per object, and thus to encode deeper hierarchies. Figure 4.4 shows two objects with the same `Genus` (green lines) but for which also DIF holds (red line, the two visual objects are different).

User feedback. We use two functions ASKGEN and ASKDIF which ask the user, when available, about `Genus` and `Differentia`. If this is not the case, they return $\textsc{Gen}(O, E)$ and $\textsc{Dif}(O, E)$, respectively. Notice how the user intervention is exploited *exactly and only* in the computation of `Genus` and `Differentia`, in order to consolidate object similarity into object identity, as from the previous section. The user feedback collected by ASKGEN

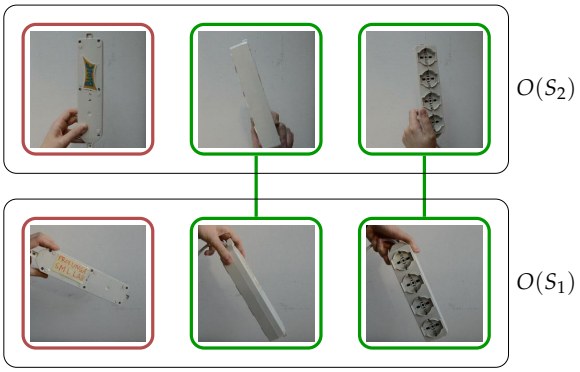


FIGURE 4.4: Two distinct objects which share the same `genusObj`. The green lines connect similar visual objects, while the identifying visual objects are highlighted in red.

and ASKDIFF is exploited by a function UPDATESIMILARITY($\mathcal{M}$) whose goal is to adjust the diversity threshold θ, see Equation (4.9), based on the knowledge available so far. The threshold is computed using the same strategy we adopted in the fully supervised algorithm shown in Section 3.1.2 each time a new supervision is provided by the user. These supervisions are stored as a set:

$$\mathcal{K} = \{\langle \delta_i, y_i \rangle \mid 1 < i < |\mathcal{K}|\}$$

where $\delta_i = d(O_i, E_i)$ is the distance between pairs of objects-encounters, coupled with a boolean value $y_i = \text{SG}(O_i, E_i)$ containing the supervision of the user. The value of θ is computed solving the following problem:

$$\theta = \underset{\lambda}{\text{argmax}} \sum_{i=1}^{|\mathcal{K}|} \mathbb{1}((\delta_i < \lambda) \oplus \neg y_i) \tag{4.16}$$

where $\mathbb{1}$ is the indicator function mapping *True* to 1 and *False* to 0, and $\oplus$ is the exclusive OR. Section 3.1.2 explains how to solve this problem efficiently.

The algorithm building the subsumption hierarchy is implemented as the infinite loop shown in Algorithm 2. This algorithm is a direct implementation of the recursive construction of objects given in Equation (4.3) via the test for object equality and subsumption as

Algorithm 2 Build subsumption hierarchy

```
1:  procedure BUILDSUBSUMPTIONHIERARCHY
2:      M ← ∅;
3:      while True do
4:          E ← PERCEIVE()
5:          if M = ∅ then
6:              ADDOBJECT(M, E)
7:              continue
8:          O ← GETMOSTSIMILAROBJECT(E, M)
9:          if ASKGEN(O, E) then
10:             if ASKDIF(O, E) then
11:                 ADDOBJECT(M, E)
12:             else
13:                 UPDATEOBJECT(M, O, O ∪ E)
14:         else
15:             ADDOBJECT(M, E)
16:         UPDATESIMILARITY(M)
```

from Equations (4.7) and (4.8) . We use a function GETMOSTSIMILAROBJECT which, given an object and a cumulative memory $\mathcal{M}$ of all the objects perceived so far, returns the most similar object. The implementation of this function is based on the consideration that there are two possible cases. In the first, that same object was previously seen and, therefore, this is the object to be selected. In the second, the object was not previously seen, in which case there may be no objects sharing visual objects (no objects with the same `genusObj`) or there may be one or more similar objects, possibly including the `genusObj`, which share the `genusObj` with the new object. Based on this intuition GETMOSTSIMILAROBJECT returns the nearest already seen instance that satisfies the similarity constraint of Equation (4.12), if existing, otherwise it returns the most similar `genusObj`, computed as described above. Notice that we make the further simplifying assumption to ask the user, via ASKGEN, only for the most similar element. Thus the model is not guaranteed to keep a hierarchy always in line with the desires of the user. Ideally one should ask for supervision for every similar object. This choice was made to limit the effort required to the user. This problem is solved in the algorithm presented in the next chapter.

For what concerns lines 9–15 of the algorithm, we have the following: (i) in Line 11,

it creates a new object because `Differentia` holds for an object with the same `Genus` (as from Equation (4.8), this is the case when subsumption holds); (ii) in Line 13, it extends an already existing object for which `Genus` holds but `Differentia` does not (as from Equation (4.7), this is the case when we have object identity); in Line 15, it creates a new object corresponding to an instance of a new `genusObj`. It is easy to see how the hierarchy satisfies, at any moment in time, the five conditions provided in the previous section. The algorithm satisfies the requirements listed at the start of the chapter: (i) the hierarchy is built autonomously and it becomes a hierarchy of classification concepts thanks to the user feedback, (ii) no assumption is made about the input objects, (iii) the algorithm learns objects never seen before and (iv) it incrementally learns how to recognize objects starting from zero.

4.6 Experiments

In the following we report first qualitative and then quantitative results in terms of capacity of the learning algorithm to recover the notions of `Genus` and `Differentia` of the user. Below, we say that the answer is correct when this is the case, incorrect otherwise. Figure 4.5 shows two cases of encounters that were correctly processed by the algorithm with no user intervention. Each column represents the sequence of steps made to process a new encounter (the visual objects in the purple box), namely perception, recognition and memorization, and the two columns represent cases giving rise to different choices made by the algorithm. The encounters already present in memory are represented by gray dashed boxes, and the corresponding objects by black boxes. A box covering visual objects from multiple objects represents the genus of that group of objects. The linked couples of blue visual objects represent items that were recognized as similar by the machine. In the left column, a new encounter is correctly recognized as having the same `Genus` of two objects

already stored in memory. The `genusObj` is updated by incorporating the visual objects of the encounter. This is all the algorithm can do, as the encounter does not have enough visual information to allow for an instance-level recognition. In the right column, a new encounter is correctly recognized as being the same as an object stored in memory. The visual objects of the encounter are added to the retrieved object, while its `genusObj` is updated with the visual objects that are found similar to it. Note how updating the object enriches its representation by including a viewpoint that was never observed before (the one showing the sockets).

Figure 4.6 shows two cases of encounters in which the algorithm made choices which are not aligned with the user perspective. In the left column, the algorithm manages to recognise the `Genus` of the new encounter, but fails to realize that the encounter is actually the same as one of the objects it has seen before. In so doing it wrongly creates a new object in memory. This error can be avoided if user feedback is available to answer an ASKDIFFERENT query (line 10 of Algorithm 2). The right column represents a case in which the new encounter was mistakenly identified as a completely different object. In this case, the availability of user supervision can prevent the algorithm from performing the wrong match (and spoiling the representation in memory of the retrieved object). The algorithm would in this case create a new object initialized with the encounter. Note however that in case the encounter was indeed an instance of an already stored object (but different from the one retrieved by the machine), or shared a `Genus` with it, asking feedback for the most similar object only as done in Algorithm 2 would not suffice to discover it (see discussion in Section 4.5.2 on the limitations of this choice). Indeed, the algorithm should progressively ask for feedback on a sequence of objects (of decreasing similarity) until the user confirms the match, which could end up being too demanding for the user. A possible solution is that asking the user to provide names for objects and genuses, thus making the mapping between substance and classification concepts explicit. This however does not

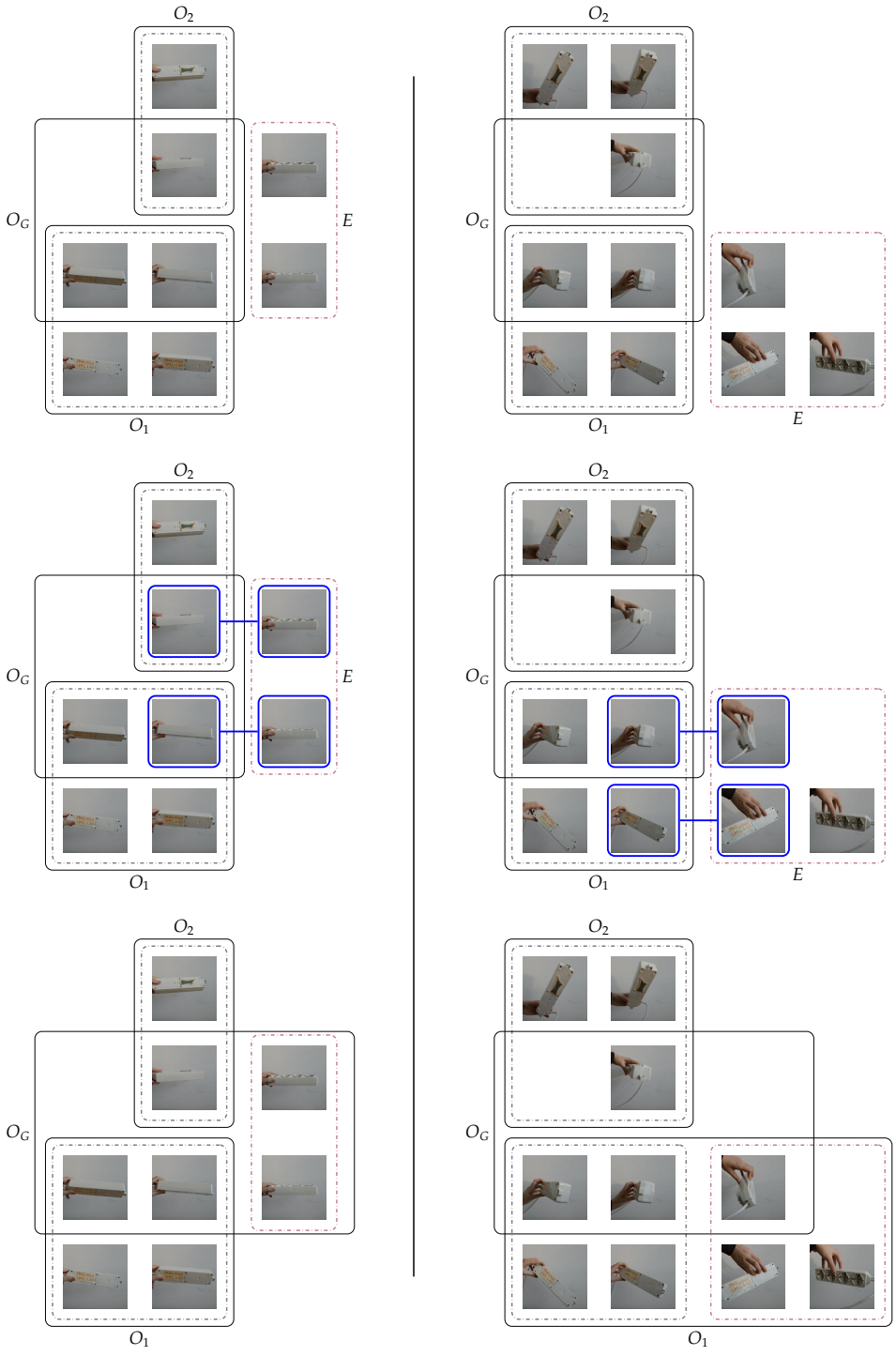


FIGURE 4.5: Examples of two correct choices made by the algorithm. The left column depicts a case in which the machine correctly identified the `Genus` of the new encounter (the encounter does not contain enough information for instance-level recognition). The second column represents a case in which the new encounter was correctly identified as an already seen object. The new visual objects were added to the matched object. In addition, the `genusObj` was updated with the subset of visual objects matching it.

solve the problem entirely, as without full supervision the memory would contain objects without names. Note also that a purely name-driven supervision cannot work, for the very reasons that have been discussed when contrasting substance concepts with classification concepts (e.g., the user could provide the name of a genus when the new encounter also has a differentia).

What described above provides a qualitative view of the behaviour of the algorithm, which largely depends on the availability of user feedback. We have also ran a quantitative evaluation showing how recognition performance over time are affected by the amount of supervision. The experiment is organized as follows. Sequences are showed one after the other and at each iteration the user provides supervision with probability α. We have run experiments for different values of α with $\alpha \in \{1.0, 0.3, 0.2, 0.1\}$ and where $\alpha = 1.0$ is the setting where supervision is always available. We have used a moving average with size fifty and stride fifteen to create the visual objects. In all settings the model is provided with a supervision for each of the first five sequences in order to bootstrap the estimation of the diversity threshold θ. We ask the model to predict `same` and `Different` at each iteration before receiving feedback from the user (if available, otherwise the algorithm prediction is used to update the memory). The results of the experiment are depicted in Figure 4.7.

Figure 4.7(a) presents the accuracy computed for the prediction of `Genus`. A prediction is correct if the new encounter shares a `Genus` with an object in memory and the algorithm retrieves the correct `genusObj`, or the algorithm correctly identifies the encounter as a completely novel object. The plotted results are computed as the mean accuracy of the prediction over two thousand different runs, each with a different order of the sequences, smoothing the curves using a moving average of length five. Surprisingly enough, in the first half of the experiment the smaller the supervision the better the accuracy. This is due to the fact that at the beginning, most sequences contain new objects, thus the more the supervision the higher the bias of the model to predict a new sequence as unseen. This bias progressively

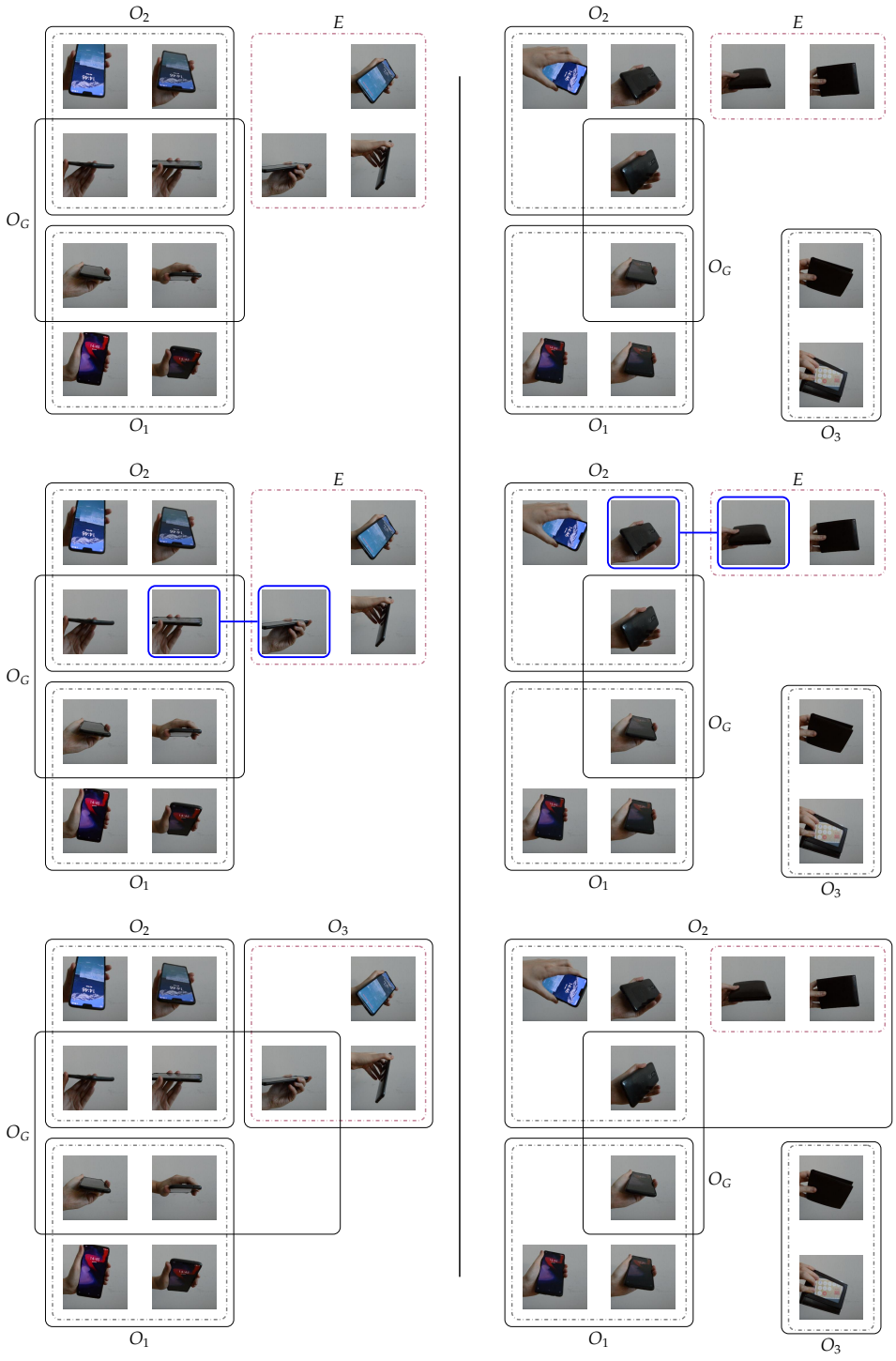


FIGURE 4.6: Examples of two incorrect choices made by the algorithm. The left column depicts a case in which the machine recognized the correct `genusObj` for the new encounter but not the correct instance. This led to the creation of a new separate object with that `genusObj`. The second column represents a case in which the new encounter (containing a wallet) was mistakenly assigned to a completely different object (a smartphone).

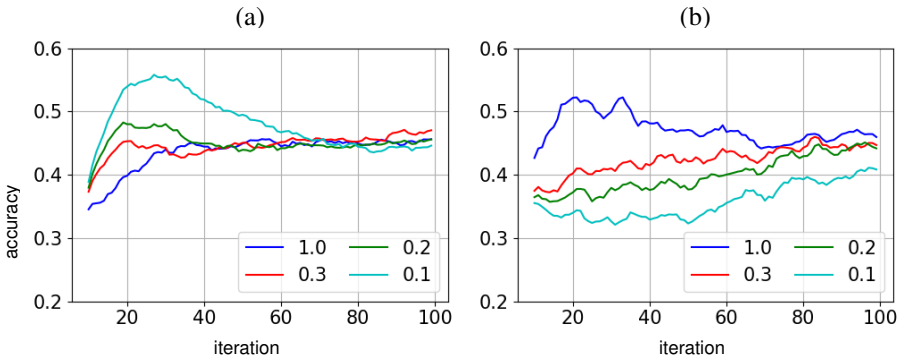


FIGURE 4.7: Accuracy of prediction for `Genus` (a) and `Differentia` (b) respectively, for increasing number of iterations and different amounts of supervision (curves for different values of α).

fades away proceeding with the experiment, and all models end up achieving similar results on average. The model with less supervision, albeit obtaining high performance in the first phase due to its bias on predicting the objects as seen, loses its advantage in the later stages of the experiments. This is due to the fact that in the first iterations the pool of objects is quite limited, but as the experiment progresses, more and more objects are added, making the prediction tasks harder.

Figure 4.7(b) shows the accuracy of the prediction of `Different`, over the subset of sequences for which the `Genus` is predicted correctly. In this case the greater the amount of supervision, the better the model is capable of recognizing whether the new sequence contains enough information to identify the correct instance. Apart for the setting with least supervision ($\alpha = 0.1$), for which the performance gap with respect to the fully supervised case stays rather large, the different models end up achieving comparable performance.

Overall, these preliminary results indicate that the algorithm is capable of progressively acquiring the notions of `Genus` and `Differentia` with reasonable accuracy despite seeing a small number of examples and receiving supervision on a fraction of them.

4.7 Summary

In this chapter we have proposed a theory and an algorithm which enables to extend the learning capabilities of the algorithm presented in the previous chapter, allowing the recognition of instances as well as classes. Albeit enabling the learning of hierarchies of two levels (e.g. instances as well as classes), the current implementations has a series of limitations compared to the algorithm we presented in the previous chapter. The algorithm described in Section 3.1.2 can determine when to elicit the supervision from the user, thus allowing for a semi supervised execution. The one presented in this chapter can cope with partial supervision, but is unable to decide *when* a supervision must be asked. Furthermore the model we presented in this chapter (as well as the one presented in Chapter 3) does not offer a clear way to build a hierarchy with more than two levels, while the theory allows for hierarchies of any depth. The Algorithm we present in the next chapter is able to overcome both these limitations, being able to handle hierarchies of any kind while having a mechanism to estimate the confidence in its own predictions, thus being able to increase new knowledge base without constant human intervention.

Chapter 5

Building the hierarchy

Chapter 4 introduced the theoretical bases to extend the learning framework described in Chapter 3 to formalize the construction of classes from instances of objects. The fact that there is no real difference between how classes and instances are handled enables to also build classes on top of other classes, obtaining what is, in the end, a full-fledged hierarchy.

The algorithm presented in Section 4.5.2 still is not able to model hierarchies deeper than two levels. This is partially due to the simplistic learning architecture inherited from the algorithms described in Chapter 3, where the prediction was entirely based on distances between representations and the unknown detection relied on a single global threshold. This chapter presents a framework that overcomes the limitations of previous ones, by substituting the simple distance-based approaches with an engine capable of building distribution probabilities for each of the objects in the hierarchy of the model. As for the model described in Algorithm 1, this framework is able to learn in a partially-supervised setting, eliciting the supervision of the user only when necessary.

The rest of the chapter is divided as follows: Section 2.4 presents the major lines of research that are related to hierarchical learning and classification. Section 5.1 details the changes that were done in implementing `Genus` and `Differentia` functions with respect to the implementations used in the previous Chapter (Section 4.5.2). Section 5.2 details the

new hierarchical framework that is the core contribution of this chapter. In Section 5.3 we present a series of experiments as well as a new dataset compatible with the setting. Finally, in Section 5.4 we present some remarks on the whole work presented in the chapter.

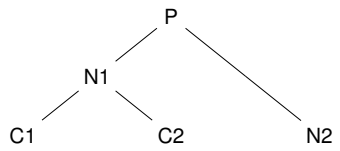
5.1 The Setting

The way in which our algorithms work is similar to the one presented in the previous chapter (Section 4.5.1). It consists of a cyclical procedure in which at each iteration a new encounter (a sequence depicting an object) is shown to the model. The model then decides whether or not to query the user for support. The interaction comes in a form of a series of queries over the `Genus` and `Differentia` of the new encounter with respect to some of the objects that were seen in the past by the algorithm (which are stored in its internal memory). Via this interaction, the user can guide the algorithm to assign the new encounter to the correct position inside the machine's knowledge base.

The dataset we used in the experiments is composed of a collection of objects organized in a perfectly balanced hierarchy of 4 levels. The visual objects are computed in the same way as in the previous chapter. The environment in which the objects were recorded is still a uniform blank background.

5.1.1 Practical considerations

The knowledge of the machine is organized following a tree-like structure, that encodes the information about the relations among all the objects it encountered, using a hierarchical tree-like structure. Each node in the hierarchy is a representation of a different object. The root represents the "thing" object, while the leaves represent single instances of objects. The internal nodes represent the various `genusObj` objects (which can be seen as classes). Two nodes that are in a particular subtree have (among many) the `genusObj` of the node



O_1	O_2	O_{ctx}	GEN(O_1, O_2, O_{ctx})	GENOF(O_1, O_2, O_{ctx})
C1	N2	*null*	$\top$	P
C1	N2	P	$\bot$	*null*
C1	C2	P	$\top$	N1
C1	C2	N1	$\bot$	*null*
C1	N1	P	$\top$	N1

FIGURE 5.1: Example of user's target hierarchy (on the left) and the output of gen and GENOF functions on a subset of target objects

that spans that particular subtree (for instance object "human" is a `genusObj` of Karl and Franz). The "thing" object is the `genusObj` of all the objects inside the hierarchy, i.e. any two objects have at least "thing" as common `genusObj`.

In order to identify and distinguish among the multiple `genusObj` between a couple of objects in the notation of this chapter we make use of a third object O_{ctx}, referred as *context*, whose role is to limit the scope of the evaluation to only knowledge contained in the context object O_{ctx}. If we imagine the two target objects (for which we are evaluating if there exists a `genusObj`) as two nodes in a tree, having both the context node as one of their ancestors, asking if there is a `genusObj` with respect to the context node means checking whether there is a descendant of the context node that is also an ancestor of the two target objects. Figure 5.1 presents an exemplar hierarchy of objects (known only by the user) together with some examples of this formalization. The function GEN(O_1, O_2, O_{ctx}) return $\top$ if there exists a `genusObj` for O_1, O_2 in the context for O_{ctx}, *i.e.* a node O_g that is a descendant of O_{ctx} and an ancestor of O_1 and O_2. If so, The function GENOF(O_1, O_2, O_{ctx}) returns the node O_g (if there are multiple candidates, the highest in the hierarchy is returned). If GEN(O_1, O_2, O_{ctx}) returns $\bot$, then no `genusObj` exists in the hierarchy between O_1, O_2 (below O_{ctx}), thus GENOF(O_1, O_2, O_{ctx}) would return $null$. If no context node is selected GEN always returns $\top$, and GENOF always returns the root of the hierarchy (the "thing" object).

For what concerns the DIF query, due to the fact that its role is to decide whether two objects are distinguishable, it does not require a context, thus it is formalized with the same

syntax used in the previous chapter. If $\textsc{Dif}(O_1, O_2)$ does not hold, the two objects are indistinguishable.

Algorithm 3 The main loop of the framework.

```
1:  procedure MAIN
2:      M ← ∅
3:      K ← ∅
4:      H ← TREE()
5:      while True do
6:          E ← PERCEIVE()
7:          if ¬SEMISUPERVISEDMODE() then
8:              O_g ← ASKNAME(r)
9:              if O_g ∉ H then
10:                 O_g ← ROOT(H)
11:             ENCOUNTERFROMGENUS(O_g, E)
12:         else
13:             SEMISUPERVISEDENCOUNTER(E)
```

5.2 The algorithm

The hierarchical framework is presented in Algorithm 3. It consists of an infinite loop that takes as input a new sequence at each iteration.

The new sequence is first forwarded to an embedding algorithm that converts the video sequence, currently encoded as a series of frames, in a collection of visual objects (i.e. the encounter E). This procedure is done by the means of a deep neural network, pretrained on a self-supervised class agnostic task [10]. The network used for this operation is the same used in the previous Chapter (discussed in Section 4.5.2).

The rest of the loop is branched in two different conditional paths, depending on the type of interaction that the algorithm must have with the user. In the first mode, the user has full control over the categorization of the newly encountered object; the role of the algorithm is to minimize the effort required to the user. This mode is useful to bootstrap the knowledge and prediction capabilities of the machine (via interaction of the user or by the means of an offline data resource) or when it is mandatory to keep perfect consistency between the

hierarchy of concepts of the user and the hierarchy of objects seen by the machine. The next phase is the actual research of the correct position in $\mathcal{H}$ in which the new information must be inserted. This research is performed by the function ENCOUNTERFROMGENUS, described in the next section. It is structured as an iterative procedure that starts from a known `genusObj` of the new encounter. In order to speed up the interaction, the user can *optionally* provide a name for the object (line 8). If the user provided a name associated to a node inside the hierarchy, the research start from that particular node, If not, the research will start from the root of $\mathcal{H}$.

The second mode enables the machine to completely bypass the user, predict its class and fully autonomously integrate the new object in its current knowledge base. This allows for a drastic reduction of the effort required by the user, albeit potentially leading to misalignment between the machine's knowledge and the user's will.

The knowledge inside the machine is organized by the means of three different data structures. The first data structure $\mathcal{H}$ represents the hierarchy of all the objects currently seen by the machine. It is encoded as a tree whose leaves are single instances of objects, and the internal nodes are the various `genusObj` used to organized the instances. The root of the tree represents the "thing" object, which represents any kind of object. Any object that will be ever seen is assumed to have at least "thing" as `genusObj`. This formulation enables the encoding of multiples layers of nested `genusObj` for each instance, each of them representing concepts that are more abstract and general the higher their position in the hierarchy. The second data structure M links each visual object extracted from each encounter with a specific node inside the hierarchy $\mathcal{H}$. This data structure enables to link the visual representations extracted from the video streams to the hierarchy of concepts hosted inside the machine's hierarchy. Its size is equal to the number of visual objects. Currently no solution is implemented to bound the memory footprint of M. But this is necessary if the pool of objects must continuously increase. A solution could be the introduction of a

selective forgetting mechanism. This option will is discussed as one of the possible future directions in the next chapter.

The last data structure K is used to store and organize the supervision received from the user and it is almost identical to the one described in Section 3.1.2. The supervision memory K is updated each time the user decides that the new encounter contains an unseen object and each time she decides that the new encounter E is the same as one object that was previously seen by the machine. The way in which this structure is updated resembles the one used in the algorithms shown in Chapter 3. Its size is equal to the number of times the user gave a supervision.

If the object was not seen before and the encounter would have been misclassified by the machine, a tuple $< p_i, \perp >$, being p_i the probability associated with the wrong prediction, is inserted in K.

If instead the user decides that the encounter contained an already seen object O_o (that was correctly predicted by the machine), then a tuple $< p_k, \top >$ is inserted in K, with p_k being the probability, estimated by the machine, that O_o and E are the same. If the machine predicts a node that is not O_o or an ancestor of O_o, no tuple is added.

The procedures that make predictions and estimates these probabilities will be discussed in detail in Section 5.2.2. The supervision memory K is used to train and refine the prediction engine.

5.2.1 Encounters

During the encounter procedure the machine, guided by the supervision of the user, walks through the hierarchy $\mathcal{H}$ searching for the right position for inserting the information provided by the new encounter. The procedure , aside from the new encounter E, takes as input a node O_g inside $\mathcal{H}$ that is considered for sure a valid `genusObj` for E (called the *current* `genusObj`). In the simplest case, O_g can be the root of $\mathcal{H}$ (*i.e.* the `genusObj`

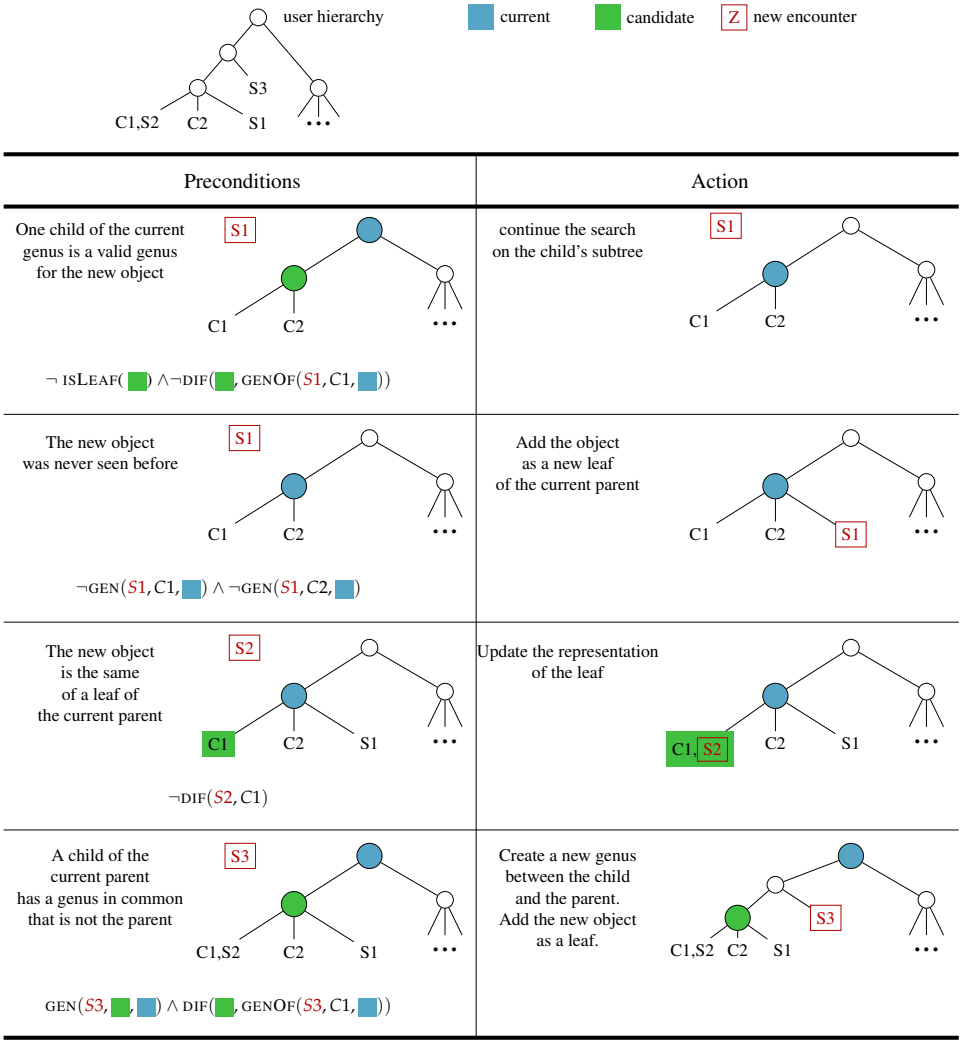


FIGURE 5.2: Representation of four possible choices that can be taken during the iterative encounter procedure. On the top the ideal hierarchy for the user is depicted. The current `genusObj` (i.e. most specific `genusObj` of the new encounter known so far) is depicted in cyan. The candidate node that is being evaluated is depicted in green. The new encounter is depicted in red. The table shows both preconditions and effects of each of the four actions. The preconditions (in the first column) are depicted as formulas. They are coupled with the portion of the hierarchy known to the machine up to that point. The effects are shown in the second column in the form of the structure of the hierarchy of the model *after* the action was taken.

"thing"). Then the machine starts moving down to find the correct node. Given O_g as the current `genusObj`, the algorithm must take one out of four possible actions. Figure 5.2 presents a graphical representation of all the four cases. On the top left it is depicted the

ideal hierarchy of concepts for the user. The new encounter is depicted in red. The current `genusObj` (i.e the node that at the previous iteration was decided as being a `genusObj` of the new encounter) is depicted in cyan, the node in the hierarchy for which the machine is asking queries is depicted in green. For each of the possible actions, each row in the table shows the preconditions that must be met (described in formulas and depicted as the portion of the hierarchy currently known to the machine) and the effect that each action has on the inner hierarchy of the machine. The four actions that can be taken are:

1. Iteratively repeat the search over one of the children of the current `genusObj`. This action requires the candidate not to be a leaf in the hierarchy and that the `genusObj` between the new encounter and one of the children of the candidate, given the current `genusObj` as the context, is equal to the candidate. This second condition is the one that determines that the candidate is a `genusObj` of the new encounter, thus enabling the candidate to become the new current `genusObj`.

2. Add the new encounter as a new object. This happens if none of the children of the current `genusObj` has a `genusObj` in common with the new encounter (except the current and all its ancestors)

3. Recognize the encounter as an object in the existing hierarchy. This happens if DIF between the candidate and the new encounter does not hold. The information in the new encounter is added to the representation of the candidate node.

4. None of the conditions above hold. The user decided that the new encounter and the candidate shares a `genusObj` that is currently *not* in the hierarchy. A new internal node is added, having as parent the current `genusObj` and as children the candidate and the new encounter (which is treated as a new object).

Algorithm 4 presents the whole encounter procedure for a new encounter E, starting from a current `genusObj` O_g. The procedure is built around two nested loops. The outer

Algorithm 4 The encounter procedure

Input : a genus g in the current hierarchy
Input : a new encoutered object E

```
1:  procedure ENCOUNTERFROMGENUS(O_g, E)
2:      continue ← ⊤
3:      while continue do
4:          continue ← ⊥
5:          found ← ⊥
6:          for O_c ∈ GETCHILDREN(O_g) do
7:              if GEN(E, O_c, O_g) then
8:                  found ← ⊤
9:                  O_n ← GEFIRSTCHILD(O_c)
10:                 if ¬ISLEAF(O_c) ∧ ¬DIF(O_c, GENOF(E, O_n, O_g)) then
11:                     O_g ← O_c
12:                     continue ← ⊤
13:                 else if ¬DIF(O_c, E) then
14:                     UPDATE(O_g, E)
15:                 else
16:                     create a new genus for O_c and E
17:                 break
18:     if ¬found then
19:         add E as new child of O_g
```

loop cycles vertically, moving one level deeper inside the hierarchy $\mathcal{H}$ at each iteration, while the inner loop cycles horizontally on each child of the current `genusObj`. The first interaction of the user is in line 7. Here the machine must determine if there exists a `genusObj` between the new encounter E and one of the children of the current `genusObj` O_g that is more specific than O_g itself, i.e. that must lie in the subtree of O_g. If this condition does not hold for any of the children of O_g , then the encounter is considered to belong to an object never seen before and is then added to $\mathcal{H}$ as a new leaf, below O_g (line 18).

If a node O_c (son of O_g) makes the query at line 7 hold, then the exploration must continue, starting from the edge that links O_g and O_c. At line 10 the procedure queries the user to verify if it is necessary to continue the search into the subtree of O_c, (*i.e.* setting $O_g = O_c$ and performing another iteration on the outer loop). Two conditions must be met for the search to proceed: first O_c can not be a leaf, due to the fact that an instance can only be the `genusObj` of itself. The second condition is that O_c must be a `genusObj` of

Algorithm 5 The predict/encounter procedure

Input : a new encouter E

1: **procedure** PREDICTENCOUNTER(E)
2: $O_g, prob \leftarrow$ PREDICT(GETROOT($\mathcal{H}$), E, 1.0)
3: **while** $\neg$GEN(E, O_g, PARENT(O_g)) **do**
4: $O_g \leftarrow$ PARENT(O_g)
5: ENCOUNTERFROMGENUS(O_g, E)

E. The second conjunct at line 10 verifies exactly this. It checks whether the `genusObj` between one of the children of O_c and E (in the context of O_g) is visually equal to O_c, i.e., if DIF between the two does not hold.

If O_c is not a `genusObj` of E, but still the user states that it has a `genusObj` (aside from O_g) in common with E, then the model asks at line 13 whether for the user DIF between O_c and E does not hold (*i.e.* the two objects are the same). If so it merges E in O_c.

If neither the condition at line 10 nor the one at line 13 hold, then the only remaining possibility is that E and O_c must be siblings but their direct parent is not O_g, but a `genusObj` that is currently missing from the machine's hierarchy $\mathcal{H}$ and must be added; this node will then have O_g as parent (E and O_c will become the grandchildren of O_g).

A refined variant of the procedure described above is presented in the algorithm 5. In this variant, the machine tries to predict the optimal point $O_g \in \mathcal{H}$ to start the encounter procedure. This approach has the potential to reduce the effort required to the user, but it is prone to errors. For this reason, this variant must check that the predicted node O_g is a genus of the new encounter E. If this is not the case the machine moves up of one level in $\mathcal{H}$ until the condition is not satisfied. The procedure that performs the prediction will be detailed in the next section.

5.2.2 Predictions

One of the pillars of the framework resides in its ability to perform Continuous Open World recognition over its ever-expanding hierarchy of categories $\mathcal{H}$. This feature is achieved via

Algorithm 6 The prediction procedure

Input : a new encouter E
Output : a (most specific) node in the hierarchy and its inclusion probability for E

```
1: function PREDICT(E)
2:     O_c ← GETROOT(H)
3:     p_r ← 1.0
4:     for v ∈ E do
5:         O_v, p_v = PREDICTVISUALOBJECT(v, GETROOT(H), 1.0)
6:         if O_c = GETROOT(H) ∨ p_r < p_v then
7:             O_c ← O_v
8:             p_r ← p_v
9:     return ⟨O_c, p_r⟩
10:
11: function PREDICTVISUALOBJECT(v, O_g, p_g)
12:     λ ← GETREJECTIONTRESHOLD(K)
13:     C ← GETCHILDREN(O_g)
14:     p_b ← max_{O_c∈C} PROBABILITY(O_c, v)
15:     O_b ← arg max_{O_c∈C} PROBABILITY(O_c, v)
16:     if p_b > λ then
17:         return PREDICTVISUALOBJECT(v, O_b, p_b)
18:     else
19:         return ⟨O_g, p_g⟩
```

a subroutine that is capable of moving through the hierarchy, identifying the best node for a new encounter, predicting the probability that the new encounter belongs to the selected node and finally deciding whether this probability is high enough to trust the prediction.

The prediction procedure is shown in Algorithm 6. It takes as input the current encounter E and returns the most specific node in the machine hierarchy to which E is predicted to belong, together with the corresponding probability. The procedure computes a prediction for each visual object $v \in E$. Remember that E is a collection of representations extracted from a stream, as discussed in Section 4.2 and depicted in Figure 4.2. Among all the predictions for each visual object, the one that is returned is the one whose probability is maximal, excluding all the predictions that return the root node (which has always probability 1.0).

The core of the procedure is the routine that computes the prediction for a single visual object, represented in Algorithm 6 as the function PREDICTVISUALOBJECT. At a high level, the prediction procedure starts at the root of the hierarchy $\mathcal{H}$ and as a first operation

decides which of the children of the "thing" node (the root) is the most likely valid prediction O_b for the visual object v. In this context a node is valid if the probability that the visual object v belongs to that node is greater than a threshold λ. If such node exists, then the procedure repeats looking at the children of O_b. The procedure terminates when either no valid node is found or the most likely valid node is a leaf. It is structured as a recursive function that takes as arguments the visual object v to be predicted, the current `genusObj` O_g (for which the algorithm decided in the previous recursion that the new object belonged to it), and the probability that v belongs to O_g, that we will call *probability of inclusion*.

The procedure starts with the computation of the threshold λ. The estimation of the correct λ is computed by solving the following optimization problem:

$$\lambda_r = \underset{\lambda}{\operatorname{argmax}} \sum_{i=1}^{|\mathcal{K}|} \mathbb{1}((p_i > \lambda) \oplus \neg y_i) \tag{5.1}$$

where $\mathbb{1}$ is the indicator function that is used to map the boolean value $\top$ to to 1 and $\bot$ to 0, and $\oplus$ is the exclusive OR. The problem is basically the same as the one presented in Equation 4.16, only applied to probabilities rather than distances.

As show in Section 3.1.2, this problem can be solved with a number of evaluations linear in the size of $\mathcal{K}$, by just keeping $\mathcal{K}$ sorted by p_i and testing all thresholds $\lambda^i = \frac{p_i + p_{i+1}}{2}$ for $i \in [0, |\mathcal{K}|]$ (where $p_0 = p_1 - \epsilon$ and $p_{|\mathcal{K}|+1} = p_{|\mathcal{K}|} + \epsilon$ for an arbitrary small ϵ).

After computing the threshold λ_r, the procedure continues by computing the probability of inclusion for all the children of O_g, and selecting the node O_b with the highest value. If the probability of inclusion of O_b is higher than the threshold λ, then the recursive step is taken and the whole procedure is computed again, starting from O_b. If the probability is lower than λ, the procedure returns O_g as the predicted node.

The estimation of the probability of inclusion can be done with any technique that is capable of estimating the probability that an element belongs to a set. Here we choose to use

Algorithm 7 Semi supervised encounter procedure

Input : a new encounter E

1: **procedure** SEMISUPERVISEDENCOUNTER(E)
2: $\lambda_l, \lambda_u \leftarrow$ GETCONFUSIONTRESHOLDS()
3: $O_g, p_c \leftarrow$ PREDICT(GETROOT($\mathcal{H}$), E, 1.0)
4: **if** $p_c > \lambda_u$ **then**
5: **if** ISLEAF(node) **then**
6: UPDATE(O_g, E)
7: **else**
8: add E as a new child of O_g
9: **else**
10: **while** $\neg$GEN(E, O_g, PARENT(O_g)) **do**
11: $O_g \leftarrow$ PARENT(O_g)
12: ENCOUNTERFROMGENUS(O_g, E)

the Extreme Value Machine (EVM) framework proposed by [79]. This framework works by selecting a subset of examples for each class it has to classify. Each of these examples, called *extreme vectors*, are associated to a Weibull distribution that is, at test phase, used to compute the probability of inclusion of a new point (based on its Euclidean distance from the closest extreme vector). Each node has its own classifier, trained by using as examples all the visual objects associated with any of the nodes in the subtree span by that node.

5.2.3 Partially-supervised Encounters

The encounter procedure described in Section 5.2.1 is based on the assumption that the user will always be queried and the correct target node inside the hierarchy $\mathcal{H}$ will always be identified. This obviously requires a lot of effort from the user. As explained in previous sections, asking or providing an entry point for the encounter procedure can lower the number of queries that the algorithm must make, but this entry point (selected by the user in line 8 of the Algorithm 3), is not guaranteed to always be present. In this section we will present a variation of the encounter procedure that enables the machine to autonomously classify and add a new encounter, bypassing the user.

The rationale of the procedure is that if the user should be queried if and only if the

machine is not secure enough of its prediction. If this is not the case, it must ask the user, as in the standard fully supervised procedure. If on the other hand it is confident enough, it can proceed to autonomously add the new knowledge inside $\mathcal{H}$. While acting unsupervised, the machine has not enough information to distinguish all the four cases presented in Figure 5.2. Specifically, it is not able to model case 4, in which it is necessary to add a new `genusObj` inside the hierarchy (and thus make it deeper). This is due to the fact that from the point of view of the algorithm a new internal node is redundant (the new object could be simply added as a new child of the parent node). For this reason the algorithm presented in this section is capable of handling only the other three cases (rows 1,2,3 in the table of Figure 5.2), namely the search step (in which the exploration continues in the subtree of the current `genusObj`), the addition of the new encounter (as an already seen object) to a node in the hierarchy and the creation of a new leaf in $\mathcal{H}$. The decision is done by computing a confusion threshold λ_u that determines the minimum probability value for which the machine is confident in the accuracy of the prediction. As in Equation 3.1 , this probability boundary is computed solving the following optimization problem:

$$\underset{\lambda_l,\lambda_u}{\text{argmax}} \quad H(\mathcal{Y}_{\lambda_l}^{\lambda_u}) - H(\mathcal{Y}^{\lambda_l}) - H(\mathcal{Y}_{\lambda_u}) \tag{5.2}$$

$$\text{subject to:} \quad \mathcal{Y}_{\lambda_l}^{\lambda_u} = \{y | \langle y, p\rangle \in \mathcal{K} \wedge \lambda_l \leq p \leq \lambda_u\}$$

$$\mathcal{Y}^{\lambda_l} = \{y | \langle y, p\rangle \in \mathcal{K} \wedge p \leq \lambda_l\}$$

$$\mathcal{Y}_{\lambda_u} = \{y | \langle y, p\rangle \in \mathcal{K} \wedge \lambda_u \leq p\}$$

$$\sum_{i=1}^{|\mathcal{K}|} \mathbb{1}(\lambda_l \leq p_i \leq \lambda_u) = \lceil \alpha |\mathcal{K}| \rceil \tag{5.3}$$

Where $\mathcal{Y}_{\lambda_l}^{\lambda_u}$ is the set of user answers related to objects with a probability within the two thresholds, while $\mathcal{Y}^{\lambda_l}$ and $\mathcal{Y}_{\lambda_u}$ are the sets related to objects below λ_l and above λ_u respectively. The function H returns the entropy of a given set. Eq. 5.3 imposes the constraint that

the user is queried on average at most with probability α, where the probability is estimated using the currently available feedback. The objective function is chosen in order to set the two thresholds in the area where the algorithm is maximally confused, adjusting the size of the area to the effort the user is willing to provide. Due to its formulation, the problem must be solved only when $\mathcal{K}$ changes, *i.e.* when the user provides a new supervision.

The semi supervised procedure is presented in Algorithm 7. After computing the confusion threshold and predicting the most specific node to which the encounter belongs, the algorithm checks whether the probability associated with the prediction is greater than the threshold λ_u. If this is the case, the algorithm autonomously adds the encounter to its hierarchy. This can happen in two ways. If the predicted node is a leaf, the encounter is considered containing an object already seen and the new information is added to that leaf. Otherwise, the object is considered as a new unseen object, and a new leaf is added as a child of the predicted node. If the probability associated with the prediction is lower than λ_u, then the user is queried to find the suitable starting point in the hierarchy, and the standard encounter procedure is executed from that.

5.3 Experiments

The dataset we used in the experiments is composed of a collection of objects organized in a perfectly balanced hierarchy of 4 levels, such that each node (except for the leaves) inside the hierarchy has 3 children, leading to a total of $3^4 = 81$ leaves. Each object was recorder 5 different times while rotated or deformed against a uniform background. The total number of video sequences is 405. The hierarchy was used to simulate the supervision of the user.

To compute the visual objects, starting from the output of the embedding network, we have used a moving average with size 50 and stride 25. In all settings the model is provided

with a supervision for each of the first 10 sequences in order to bootstrap the estimation of the thresholds.

The experiments are performed by showing the sequences to the machine, one after the other. Before the supervision the models are queried for a prediction on the current object, which is then used to build the plots shown in the following sections. The whole operation is performed 100 times, each time changing the order in which the sequences were shown to the model. The results are averaged in order to obtain more robust estimates of performance measures.

The user is simulated by an agent that provides supervision to the model by comparing the hierarchy of the dataset against the hierarchy of the machine, and replying to the queries of the algorithm accordingly. As the user is simulated in our experiments, we skip the step of asking the user to suggest a name for the new encounter (as in line 8 of Algorithm 3); the encounter procedure always start at the "thing" node (the root of $\mathcal{H}$) or at a node suggested by the machine. All the code used to implement and evaluate the algorithm, as well as the dataset, is available in a early release form[1].

5.3.1 Full supervision

The first series of experiments were conducted in a fully supervised setting. The goal of the algorithm in this setting is to aid the user in the task of categorizing all objects she encounters, by suggesting the correct point in the hierarchy $\mathcal{H}$ in which each object should be placed.

In this setting, the algorithm has the sole role of speeding up the identifications of the correct location of the new encounter int the hierarchy, thus minimizing the effort of the user. The hierarchy of objects that will be constructed at the end, as well as the intermediate ones, will always be consistent with the ground truth. As explained in section 5.2.1, the

[1] Early release: https://github.com/lucaerculiani/hierarchical-objects-learning

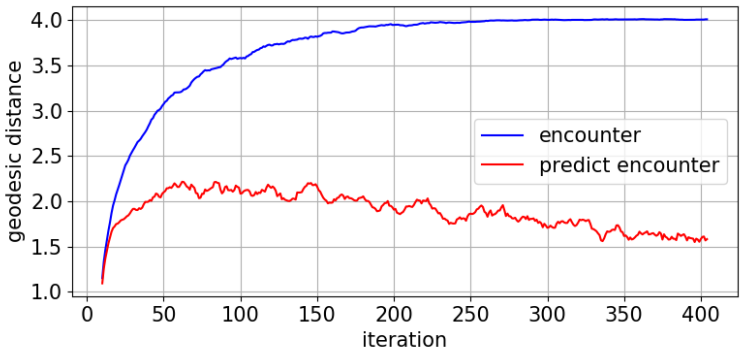


FIGURE 5.3: Average geodesic distance between the predicted and the correct node in a setting with complete supervision, during the course of the experiment. The blue line represents the *naive* model (that always predict the root node), while the red line represents the model described in algorithm 5.

machine aids the user by predicting the starting node, in $\mathcal{H}$, from which the user navigates the hierarchy in search of the correct node. The more closer the prediction of the machine (to the correct node), the lower the effort of the user.

The performance metric used in this set of experiments is the *geodesic distance* (the number of edges in the shortest path) between the node predicted by the machine and the target node selected by the user. If the algorithm predicts the correct node, the cost is 0, the more distant the prediction in $\mathcal{H}$, the greater the cost. Even if this measure is affected by the size of the hierarchy (the deeper the tree, the greater the average distance between couples of nodes), due to the fact that the evolution of the hierarchy is completely guided by the user, any fully-supervised model has always its hierarchy updated in the same way. This fact keeps this performance measure unbiased in the context of this experiment.

Figure 5.3 presents the average geodesic distance between predicted and correct node for a model (the red line) using Algorithm 5, compared with a *naive* algorithm (in blue) that always suggests the root of $\mathcal{H}$ as a starting node. The objects were presented in a random order, and the experiments were repeated 100 times, averaging the results at each iteration.

This experiment can be seen as an ablation study to determine whether the prediction framework is actually useful to the user. A badly behaving predictor could in principle do worse than the naive algorithm, by predicting a node in a subtree that is more distant than the root node.

Our model substantially outperforms the naive algorithm in this fully-supervised task. After an initial phase in which the cost increases due to the rapid expansion of the hierarchy, the average geodesic distance starts decreasing. This is due to the fact that the algorithm starts accumulating more and more encounters for each of the objects, and this enables it to more accurately model each class. Meanwhile, as expected, the naive algorithm converges toward an average cost that is equal to the average distance between the root nodes and the leaves (the hierarchy of the dataset is composed by four levels).

5.3.2 Partial supervision

The next series of experiments are aimed at evaluating the performance of the machine in a partially-supervised setting, in which the machine is required to classify and add to its knowledge base new encounters without asking to the user, whenever possible. This setting aims to evaluate the trade-offs between the loss in recognition performance and the amount of times the user is spared from the task of actively classifying the new encounter.

Due to the intermittent supervision of the user, there is no guarantee that the hierarchy $\mathcal{H}$ remains consistent with the desires of the user. If the machine makes a mistake that is not corrected, the hierarchy will lose consistency with respect to the one of the user.

Different variants of the algorithm will thus produce different hierarchies. This fact limits the applicability of the geodesic distance as the performance metric of choice. In a balanced tree of depth n, the maximum geodesic distance between nodes is the one between two leaves at the opposite sides of the tree, and it is equal to $2n$. Due to this fact, the metric

tends to downplay errors made on shallow hierarchies, thus penalizing deeper hierarchies, over things being equal.

In order to avoid this bias, as suggested by [89], we used the *hierarchical F-score* (hF) as performance measure. The hF-score, analogously to the standard f-score, is defined from two measures called hierarchical precision (hP) and recall (hR). Given a set of i couples of predicted and true node, and being T_i the set of all nodes in the path from the root to the true target node (root and target node included), and P_i the set of all nodes in the path from the root to the predicted node node (extremes included), hP, hR, and hF score are defined as:

$$hP = \sum_i \frac{|T_i \cap P_i|}{|P_i|} \tag{5.4}$$

$$hR = \sum_i \frac{|T_i \cap P_i|}{|T_i|} \tag{5.5}$$

$$hF = \frac{2 * hP * hR}{hP + hR} \tag{5.6}$$

Figure 5.4(a) presents the average hF score at each iteration, averaged over 100 repetitions, of several models using the partially-supervised encounter procedure described in Algorithm 7. The plot presents different instances of the same model with different levels of α (0.95 in red, 0.9 in green and 0.8 in cyan), compared with the fully-supervised model of Algorithm 5. For the same models, Figure 5.5(b) presents the probability of requesting supervision, estimated counting the fraction of times in which the model required supervision over the 100 runs.

As expected, the best performance are obtained by the fully supervised model, and decrease in the amount of supervision translates in a decrease of hF score. However, even when the amount of supervision is provided in less than half of the encounters, as it is the case with the model with $\alpha = 0.8$, the model is still able to converge to a stationary value of hF score that is 66% of the fully supervised hF score. According to the needs of the setting

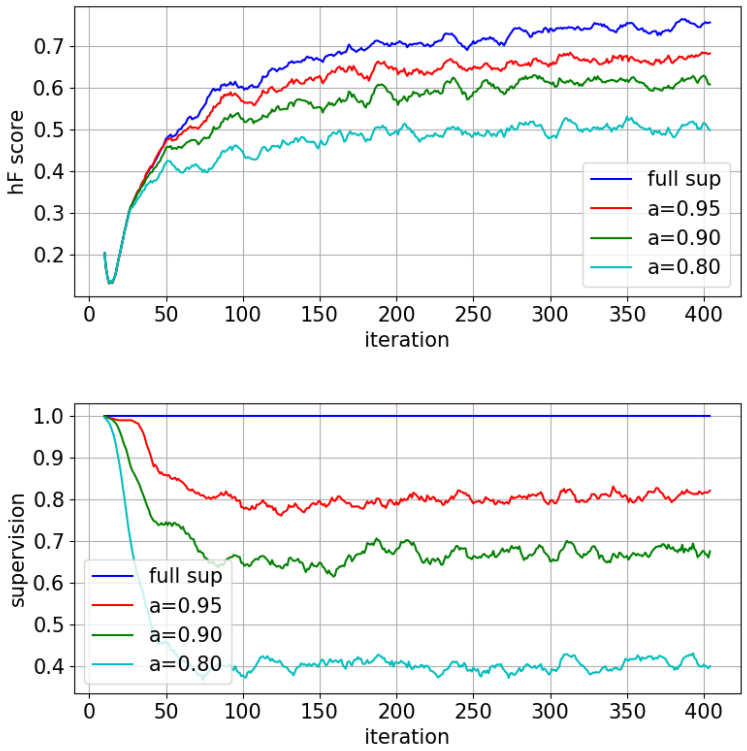


FIGURE 5.4: Plot (a) presents the average hierarchical F-score of a model that receives full supervision, in blue, compared with a series of partially-supervised models with different values of α. Plot (b) presents the average estimated probability of receiving supervision, at each iteration during the experiment.

the model can be tailored to reach certain levels of performance (at the cost of an increase in the effort of the user) by tuning the α parameter.

5.3.3 A developmental approach

The inquiries made in Chapter 3 about the order in which the new objects were introduced to the machine led to the conclusion that a developmental approach, in which the new objects were introduced gradually, aided the machine. In that setting this was implemented by keeping track of all the objects the machine had already seen, and at each new encounter showing an unseen object with a probability equal to 0.3. In this way the introduction of new object was spread more evenly through the entire experiment.

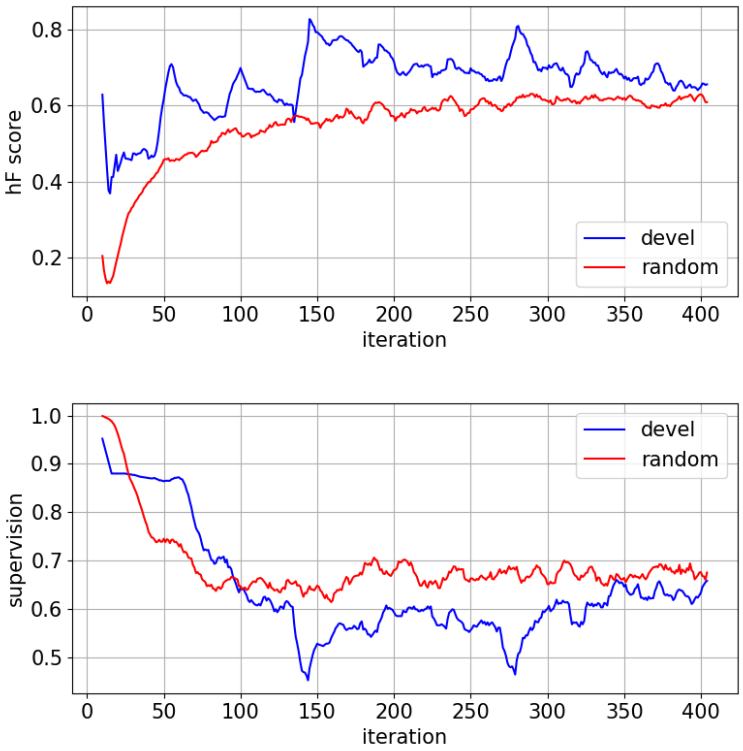


FIGURE 5.5: Plot a presents the average hierarchical F-score of two partially-supervised models, exposed to the objects in a random order (in blue), compared with one exposed to the objects using the `devel` policy. Plot b presents the average estimated probability of receiving supervision, at each iteration during the experiment.

The experiments seen so far in this chapter use a random order instead. The effect of this is that at the start of the experiment the probability of seeing a new object at each encounter is quite high. In order to verify whether the findings in Chapter 3 carry over to the hierarchical case, we also implemented a developmental experimental setting (referred to as `devel`) and compared the performance of the algorithm in that setting with those achieved in the random setting. Due to the hierarchical nature of our dataset, we decided to use a different strategy than the one used in Chapter 3 in implementing the developmental setting. At each iteration, the next encounter that is presented is the one whose geodesic distance with respect to the last encountered object is minimal. With this policy, the machine is initially exposed to a minimal subtree of objects in the hierarchy of the dataset, and then

gradually new objects coming from sibling subtrees are introduced.

Figure 5.5(a) presents the hF score of our partially-supervised algorithm in the random (red) and `devel` (blue) setting respectively. For the sake of a fair comparison, we set the value of alpha in each setting in order for the two settings to get a similar level of supervision ($\alpha = 0.95$ for random and 0.9 for `devel`), as shown in figure 5.5(b). Despite requiring a slightly lower amount of supervision on average (64% for `devel` vs 69% for random), the performance of the `devel` setting are consistently higher than those of the random one across all iterations, with an average hF score of 0.66 as compared with 0.54 achieved by the random setting. The zig-zag nature of the `devel` line in Figure 5.5(a) is due to the order of the objects in this setting. In particular, these peaks arise when all objects in the current subtree have already been seen and a new object from a different subtree is shown to the model. This momentarily boosts the recognition capabilities of the model due to the fact that the new object is completely different from the previous encounters, making it easier to classify. As more and more objects are shown, these fluctuations of performance tend to fade.

The results of this set of experiments confirm what was postulated in Chapter 3. In this type of continuous Open World scenarios, the policy that is used to introduce new knowledge has a strong impact on the performance of the model. A gradual introduction of new objects allows the algorithm to build better class representations faster, that in turn boosts its performance both in the short term (when the knowledge is still partial) and in the long run.

5.4 Summary

In this chapter we detailed a first complete implementation of an algorithm that is capable of performing the task of continuously learning hierarchies of objects closely following the

desires of a given human. The algorithm extracts information from a continuous stream of visual data, and does not require a fixed ontology of concepts, while at the same time being able to recognize and add previously unknown objects even without supervision.

The experiments presented in Section 5.3 demonstrate the capability of the framework, and show that, consistently with the findings obtained in Chapter 3, the order in which new and old knowledge is interleaved during the learning process can have a substantial role in boosting performance. Many lines of research still remain open, and many aspects of the current architecture can be refined with further research. Some of these directions for future works will be detailed in the next chapter.

Chapter 6

Conclusion

In the last decade the whole field of Machine Learning has witnessed an explosive surge in popularity, mainly led by the success of Deep Neural Networks. At the same time new and ever-increasing data sources were made available, in order to accommodate the training requirements of this kind of architectures. Over the last few years many have started questioning this growth in the complexity of neural architectures. Many new tasks arose, whose goal being explicitly to try to overcome the dependency of these algorithms on huge immutable data sources. From Few-Shots to Open World learning, many new challenges were proposed and corresponding technique were developed.

Despite this surge of attention, no model found in literature was able to learn in a setting which is similar to the one in which humans learn. Moreover, these algorithms rely on a kind of information that is structured in the same static class-based way. Example are still categorized in fixed classes, while humans perceive the world as a continuous stream of stimuli, and build a semantic meaning to what they see only afterwards.

The focus of this thesis has been to develop techniques to increase the similarity between the learning setting of the machine and the one of its user. We tackled this challenge in a series of steps, that lead us to build a full-fledged Continuous Open World learning algorithm without giving away the basic properties we identified during the course of the

last three years.

In Chapter 3 we started postulating a first approach to the task. The setting we proposed can be seen as a combination of the Open World and Few-Shots learning tasks. In order to avoid the pitfalls of the class-based supervision, we decided to limit the recognition to just instances of objects. The fact that an object is equal to itself always holds, without requiring to semantically interpret what that object is (i.e labeling it). In that line of research we decided to experiment with some policies to control the rate at which new objects were shown to the algorithm, and we found out that a developmental strategy is indeed beneficial to boost the performance of the model while reducing the supervision effort of the user.

In order to extend our framework beyond the recognition of instances, in Chapter 4 we developed a new theory to allow the creation of class representations without the need to introduce and ontology-based labeling system. Our theory revolves once again around the perception, defining instances (and classes) of objects in terms of what made them visually distinguishable (and indistinguishable) from other objects. Together with the theory, we provided a first implementation in the form of an algorithm that, borrowing from the structure of the algorithm developed in Chapter 3, is capable of recognizing hierarchies of objects of two levels (i.e. instances and classes). Due to the fact that the core of the algorithm is shared with its ancestor, some compromises had to be done. The new algorithm loses the ability to elicit supervision from the user and is more susceptible to input noise.

Chapter 5 detailed our first model that is able to dynamically learn an indefinitely deep hierarchy of objects, while keeping, as it was for the algorithm in Chapter 3, the ability to actively elicit supervision. Among the many improvements, this was made possible by a newly redesign interaction with the user (that still happens inside the framework defined in Chapter 4), as well as the switch from a nearest-neighbor prediction to a model that estimates probability distributions for each of the nodes in the hierarchy of objects. A series of experiments showcases the capabilities of the model, while confirming the beneficial role

of a developmental policy in boosting its performance.

6.1 Future lines of research

During the course of the last three years, we have been developed a framework that learns and reasons in a setting that is the closest in the field to the setting in which humans learn. Still, the road to human-level learning capabilities is quite long. Due to the limited time of a PhD scholarship, many lines of research we envisioned still remain unexplored. In this Section we will hint the most prominent ones.

Experiments with real users

In all the variations of the framework we developed, the continuous interaction of the machine with the user has a central role. The key missing element in our evaluation procedures are experiments with real users. These would be especially useful to test the ability of our most complete model (described in Chapter 5), to adapt and capture the different user's needs, in particular concerning the structure of the hierarchy they are interested in. Another useful experiment would be comparing the predictions of the machine with the one of the user in all the cases in which the encounter does not contain enough information to associate it with a leaf in the hierarchy (e.g person that is too far to be identified).

Knowledge consolidation and forgetting

One key element that empowers humans is the ability to continuously learn new information without interfering with their previous knowledge. This does not mean that the knowledge of a person always increases monotonically. Instead, selectively forgetting useless information is an integral part of the learning process. We think that our models too could benefit from a forgetting mechanism. This could help both to boost the recognition performance of

the subset of objects that are of interest for the user, and allow to more easily scale with the number of recognized objects by removing duplicate information.

Learning visual features

The current architecture makes use of a pretrained Neural Network to extract higher level representations from raw visual data (the sequence of frames). This technique is based on a model trained to match general visual features extracted from a large collection of random images (at least for the algorithm described in Chapter 4 and 5). This decision was made due to the difficulty of training an embedding network using few (and highly correlated) images. Future research on this topic could enable the specialization of the embedding network to the environment in which the machine operates, as well as to the subset of objects that are of most interest for the user.

Horizontal relations in the hierarchy of objects

The current theory that supports our approach of learning hierarchies of objects accounts for vertical relations among nodes (i.e. the `genusObj`), but does not specify any kind of framework to capture horizontal relations among concepts. An example of this is the *part-of* property. This would enable to lay a well-defined foundation to model for instance objects that are composed by groups of objects, arranged in a meaningful way. This could open the possibility to tackle other problems too, like the distinction between the target object and the background that appears around it.

Information fusion

The relation between the perception of the human and the one of the machine is a key aspect of this research. Due to the novelty of this work, we have been able to focus only on visual perception. An important step forward would be shifting the focus towards including other

forms of perception, such as the acoustic one, in order to exploit information across multiple senses.

Many other questions still remain open. How can we cope with the natural noise and randomness that characterize the visual appearance of objects (and that if not accounted for can disrupt the learning process)? How can the past information, obtained in a time in which the model had not received enough training, coexist with newer (and more refined) object representations? How should the machine deal with inconsistent users? We leave them to the reader, hoping to foster further research on this ambitious and challenging goal.

Bibliography

[1] Steven Abney. “Part-of-speech tagging and partial parsing”. In: *Corpus-based methods in language and speech processing*. Springer, 1997, pp. 118–136.

[2] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. “Youtube-8m: A large-scale video classification benchmark”. In: *arXiv preprint arXiv:1609.08675* (2016).

[3] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. “Gene ontology: tool for the unification of biology”. In: *Nature genetics* 25.1 (2000), pp. 25–29.

[4] Eden Belouadah and Adrian Popescu. “Il2m: Class incremental learning with dual memory”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 583–592.

[5] A. Bendale and T. E. Boult. “Towards open set deep networks”. In: *CVPR*. 2016, pp. 1563–1572.

[6] A. Bendale and T. E. Boult. “Towards open world recognition”. In: *CVPR*. 2015, pp. 1893–1902.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. “Curriculum Learning”. In: *ICML*. ICML ’09. 2009, pp. 41–48.

[8] Helyane Bronoski Borges and Julio Cesar Nievola. "Multi-label hierarchical classification using a competitive neural network for protein function prediction". In: *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2012, pp. 1–8.

[9] C. Braun, U. Heinz, R. Schweizer, K. Wiech, N. Birbaumer, and H. Topka. "Dynamic organization of the somatosensory cortex induced by motor activity". In: *Brain* 124.11 (2001), pp. 2259–2267.

[10] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin. "Unsupervised pre-training of image features on non-curated data". In: *ICCV*. 2019, pp. 2959–2968.

[11] Ricardo Cerri, Rodrigo C Barros, André CPLF de Carvalho, and Yaochu Jin. "Reduction strategies for hierarchical multi-label classification in protein function prediction". In: *BMC bioinformatics* 17.1 (2016), p. 373.

[12] E. Clerkin, E. Hart, J. Rehg, C. Yu, and L. Smith. "Real-world visual statistics and infants' first-learned object names". In: *RSTB* 372 (2017).

[13] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. "Hierarchical clustering: Objective functions and algorithms". In: *Journal of the ACM (JACM)* 66.4 (2019), pp. 1–42.

[14] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *CVPR*. IEEE. 2009, pp. 248–255.

[15] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. "Learning without memorizing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5138–5146.

[16] T. J. Draelos, N. E. Miner, C. C. Lamb, J. A. Cox, C. M. Vineyard, K. D. Carlson, W. M. Severa, C. D. James, and J. B. Aimone. "Neurogenesis deep learning: Extending deep networks to accommodate new classes". In: *IJCNN*. IEEE. 2017, pp. 526–533.

[17] Susan Dumais and Hao Chen. "Hierarchical classification of web content". In: *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. 2000, pp. 256–263.

[18] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. "Adversarial Continual Learning". In: *ECCV* (2020).

[19] L. Erculiani, F. Giunchiglia, and A. Passerini. "Continual egocentric object recognition". In: *ECAI* (2020).

[20] Andrea Esuli, Tiziano Fagni, and Fabrizio Sebastiani. "Boosting multi-label hierarchical text categorization". In: *Information Retrieval* 11.4 (2008), pp. 287–313.

[21] C. Finn, P. Abbeel, and S. Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *ICML*. 2017, pp. 1126–1135.

[22] Mattia Fumagalli, Gabor Bella, and Fausto Giunchiglia. "Towards understanding classification and identification". In: *Pacific Rim International Conference on Artificial Intelligence* (2019), pp. 71–84.

[23] P. Földiák. "Learning Invariance from Transformation Sequences". In: *Neural Computation* 3.2 (1991), pp. 194–200.

[24] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. "Sweetening ontologies with DOLCE". In: *International Conference on Knowledge Engineering and Knowledge Management*. Springer. 2002, pp. 166–181.

[25] A. Gepperth and C. Karaoguz. "A bio-inspired incremental learning architecture for applied perceptual problems". In: *Cognitive Computation* 8.5 (2016), pp. 924–934.

[26] M. Gini, N. Agmon, F. Giunchiglia, S. Koenig, and K. Leyton-Brown. "Artificial Intelligence in 2027". In: *AI Matters* 4.1 (2019).

[27] F. Giunchiglia, K. Batsuren, and A. A. Freihat. “One World–Seven Thousand Languages”. In: *Proceedings 19th International Conference on Computational Linguistics and Intelligent Text Processing*. 2018, pp. 18–24.

[28] F. Giunchiglia, Khuyagbaatar Batsuren, and Gabor Bella. “Understanding and Exploiting Language Diversity.” In: *IJCAI*. 2017, pp. 4009–4017.

[29] F. Giunchiglia and M. Fumagalli. “Concepts as (Recognition) Abilities.” In: *FOIS*. 2016, pp. 153–166.

[30] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. “An empirical investigation of catastrophic forgetting in gradient-based neural networks”. In: *ICLR*. 2014.

[31] Nicola Guarino. “The role of identity conditions in ontology design”. In: *International Conference on Spatial Information Theory*. Springer. 1999, pp. 221–234.

[32] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *CVPR*. 2016, pp. 770–778.

[33] D. Hebb. “The organization of behavior. 1949”. In: *New York Wiely* 2.7 (2002), p. 8.

[34] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[35] E. Hirsch. *The concept of identity*. Oxford University Press, 1982.

[36] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[37] Rie Johnson and Tong Zhang. “Effective use of word order for text categorization with convolutional neural networks”. In: *arXiv preprint arXiv:1412.1058* (2014).

[38] René Traoré Kalifou, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Natalia Diaz-Rodriguez, and David Filliat. "Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer". In: *ICML Workshop on Multi-Task and Lifelong Learning*. 2019.

[39] N. Kamra, U. Gupta, and Y. Liu. "Deep Generative Dual Memory Network for Continual Learning". In: *arXiv preprint arXiv:1710.10368* (2017).

[40] R. Kemker and C. Kanan. "Fearnet: Brain-inspired model for incremental learning". In: *ICLR*. 2018.

[41] Salabat Khan, Abdul Rauf Baig, and Waseem Shahzad. "A novel ant colony optimization based single path hierarchical classification algorithm for predicting gene ontology". In: *Applied Soft Computing* 16 (2014), pp. 34–49.

[42] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. "Overcoming catastrophic forgetting in neural networks". In: *PNAS* (2017), p. 201611835.

[43] S. Kirstein, A. Denecke, S. Hasler, H. Wersing, H. Gross, and E. Körner. "A vision architecture for unconstrained and incremental learning of multiple categories". In: *Memetic Computing* 1.4 (2009), p. 291.

[44] G. Koch. "Siamese neural networks for one-shot image recognition". In: *ICML Deep Learning Workshop*. 2015.

[45] Daphne Koller and Mehran Sahami. *Hierarchically classifying documents using very few words*. Tech. rep. Stanford InfoLab, 1997.

[46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[47] Zhenzhong Kuang, Jun Yu, Zongmin Li, Baopeng Zhang, and Jianping Fan. "Integrating multi-level deep learning and concept ontology for large-scale visual recognition". In: *Pattern Recognition* 78 (2018), pp. 198–214.

[48] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. "Recurrent convolutional neural networks for text classification". In: *Twenty-ninth AAAI conference on artificial intelligence*. 2015.

[49] E. Lalumera. "Concepts are a functional kind". In: *Behavioral and Brain Sciences* 33.2-3 (2010), pp. 217–218.

[50] W. Li, X. Zhu, and S. Gong. "Harmonious attention network for person re-identification". In: *CVPR*. 2018, pp. 2285–2294.

[51] Z. Li and D. Hoiem. "Learning without forgetting". In: *TPAMI* (2017).

[52] X. Liu, W. Liu, H. Ma, and H. Fu. "Large-scale vehicle re-identification in urban surveillance videos". In: *ICME*. IEEE. 2016, pp. 1–6.

[53] V. Lomonaco and D. Maltoni. "CORe50: a New Dataset and Benchmark for Continuous Object Recognition". In: *CoRL*. Vol. 78. PMLR, 2017, pp. 17–26.

[54] Xiang Long, Chuang Gan, Gerard De Melo, Jiajun Wu, Xiao Liu, and Shilei Wen. "Attention clusters: Purely attention based local feature integration for video classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7834–7843.

[55] G. Macdonald and D. Papineau. *Teleosemantics*. Oxford University Press, 2006.

[56] A. Martin and L. L. Chao. "Semantic memory and the brain: structure and processes". In: *Current opinion in neurobiology* 11.2 (2001), pp. 194–201.

[57] J. L. McClelland, B. L. McNaughton, and R. C. O'reilly. "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." In: *Psychological review* 102.3 (1995), p. 419.

[58] N. McLaughlin, J. Martinez del Rincon, and P. Miller. "Recurrent convolutional network for video-based person re-identification". In: *CVPR*. 2016, pp. 1325–1334.

[59] Umberto Michieli and Pietro Zanuttigh. "Knowledge distillation for incremental learning in semantic segmentation". In: *Computer Vision and Image Understanding* 205 (2021), p. 103167.

[60] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. "Introduction to WordNet: An on-line lexical database". In: *International journal of lexicography* 3.4 (1990), pp. 235–244.

[61] R. G. Millikan. "A more plausible kind of "recognitional concept"". In: *Philosophical Issues* 9 (1998), pp. 35–41.

[62] R. G. Millikan. "Biosemantics". In: *The journal of philosophy* 86.6 (1989), pp. 281–297.

[63] R. G. Millikan. *Language: A biological model*. Oxford University Press on Demand, 2005.

[64] R. G. Millikan. *Language, thought, and other biological categories: New foundations for realism*. MIT press, 1984.

[65] R. G. Millikan. *On clear and confused ideas: An essay about substance concepts*. Cambridge University Press, 2000.

[66] R. G. Millikan. *Varieties of meaning: the 2002 Jean Nicod lectures*. MIT press, 2004.

[67] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. "A simple neural attentive meta-learner". In: *ICLR*. 2018.

[68] D. K. Naik and R. Mammone. "Meta-neural networks that learn by learning". In: *IJCNN*. Vol. 1. IEEE. 1992, pp. 437–442.

[69] T. Nakamura, K. Sugiura, T. Nagai, N. Iwahashi, T. Toda, H. Okada, and T. Omori. "Learning novel objects for extended mobile manipulation". In: *Robotic Systems* 66.1-2 (2012), pp. 187–204.

[70] H. (ed.) Noonan. *Identity*. Dartmouth, Aldershot USA, 1993.

[71] L. Nuo and J. J. DiCarlo. "Unsupervised natural experience rapidly alters invariant object representation in visual cortex". In: *Science* 321.5895 (2008). PMID: 18787171, 1502–1507.

[72] M. Oliveira, L. S. Lopes, G. H. Lim, S. H. Kasaei, A. M. Tomé, and A. Chauhan. "3D object perception and perceptual learning in the RACE project". In: *Robotics and Autonomous Systems* 75 (2016), pp. 614–626.

[73] Fernando EB Otero, Alex A Freitas, and Colin G Johnson. "A hierarchical classification ant colony algorithm for predicting gene ontology terms". In: *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*. Springer. 2009, pp. 68–79.

[74] Fernando EB Otero, Alex A Freitas, and Colin G Johnson. "A hierarchical multi-label classification ant colony algorithm for protein function prediction". In: *Memetic Computing* 2.3 (2010), pp. 165–181.

[75] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. "Continual Lifelong Learning with Neural Networks: A Review". In: *arXiv preprint arXiv:1802.07569* (2018).

[76] S. Ravi and H. Larochelle. "Optimization as a model for few-shot learning". In: (2017).

[77] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection". In: *CVPR*. 2016, pp. 779–788.

[78] X. Ren and M. Philipose. "Egocentric recognition of handled objects: Benchmark and analysis". In: *CVPR Workshop*. IEEE. 2009, pp. 1–8.

[79] E. M. Rudd, L. P. Jain, W. J. Scheirer, and T. E. Boult. "The extreme value machine". In: *TPAMI* 40.3 (2018), pp. 762–768.

[80] Miguel E Ruiz and Padmini Srinivasan. "Hierarchical text categorization using neural networks". In: *Information retrieval* 5.1 (2002), pp. 87–118.

[81] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *IJCV* 115.3 (2015), pp. 211–252.

[82] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (2016).

[83] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. "Meta-learning with memory-augmented neural networks". In: *ICML*. 2016, pp. 1842–1850.

[84] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. "Toward open set recognition". In: *TPAMI* 35.7 (2013), pp. 1757–1772.

[85] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. "Progress & compress: A scalable framework for continual learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4528–4537.

[86] Or Sharir, Barak Peleg, and Yoav Shoham. "The Cost of Training NLP Models: A Concise Overview". In: *arXiv preprint arXiv:2004.08900* (2020).

[87] Y. Shen, T. Xiao, H. Li, S. Yi, and X. Wang. "Learning deep neural networks for vehicle re-id with visual-spatio-temporal path proposals". In: *ICCV*. IEEE. 2017, pp. 1918–1927.

[88] H. Shin, J. K. Lee, J. Kim, and J. Kim. "Continual learning with deep generative replay". In: *NIPS*. 2017, pp. 2990–2999.

[89] Carlos N Silla and Alex A Freitas. "A survey of hierarchical classification across different application domains". In: *Data Mining and Knowledge Discovery* 22.1-2 (2011), pp. 31–72.

[90] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. "Calibrating CNNs for Lifelong Learning". In: *Advances in Neural Information Processing Systems* 33 (2020).

[91] D. Skočaj, M. Kristan, A. Vrečko, M. Mahnič, M. Janíček, G. M. Kruijff, M. Hanheide, N. Hawes, T. Keller, M. Zillich, et al. "A system for interactive learning in dialogue with a tutor". In: *IROS*. IEEE. 2011, pp. 3387–3394.

[92] Arnold Smeulders, Marcel Worring, Simone Santini, and Ramesh Jain. "Content-based image retrieval at the end of the early years". In: *IEEE Transactions on PAMI* 22.12 (2000), pp. 1349–1380.

[93] L. B. Smith and L. K. Slone. "A Developmental Approach to Machine Learning?" In: *Frontiers in Psychology* 8 (2017), p. 2124.

[94] J. Snell, K. Swersky, and R. Zemel. "Prototypical networks for few-shot learning". In: *NIPS*. 2017, pp. 4080–4090.

[95] E. Triantafillou, R. Zemel, and R. Urtasun. "Few-shot learning through an information retrieval lens". In: *NIPS*. 2017, pp. 2255–2265.

[96] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra. "Matching networks for one shot learning". In: *NIPS*. 2016, pp. 3630–3638.

[97] T. Wang, S. Gong, X. Zhu, and S. Wang. "Person re-identification by video ranking". In: *ECCV*. Springer. 2014, pp. 688–703.

[98] H. Wersing, S. Kirstein, M. Götting, H. Brandl, M. Dunn, I. Mikhailova, C. Goerick, J. Steil, H. Ritter, and E. Körner. "Online learning of objects in a biologically motivated visual architecture". In: *Neural Systems* 17.04 (2007), pp. 219–230.

[99] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, and Bogdan Raducanu. "Memory replay gans: Learning to generate new categories without forgetting". In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 5962–5972.

[100] Y. Wu, Y. Lin, X. Dong, Y. Yan, W. Ouyang, and Y. Yang. "Exploit the unknown gradually: One-shot video-based person re-identification by stepwise learning". In: *CVPR*. 2018, pp. 5177–5186.

[101] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 305–321.

[102] J. Xu, R. Zhao, F. Zhu, H. Wang, and W. Ouyang. "Attention-Aware Compositional Network for Person Re-identification". In: *CVPR*. 2018.

[103] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. "Lifelong Learning with Dynamically Expandable Networks". In: *ICLR*. 2018.

[104] F. Zenke, W. Gerstner, and S. Ganguli. "The temporal paradox of Hebbian learning and homeostatic plasticity". In: *Neurobiology* 43 (2017), pp. 166–176.

[105] F. Zenke, B. Poole, and S. Ganguli. “Continual Learning Through Synaptic Intelligence”. In: *ICML*. 2017, pp. 3987–3995.

[106] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. “Class-incremental learning via deep model consolidation”. In: *WACV*. 2020, pp. 1131–1140.

[107] Yu Zheng, Qiuyu Chen, Jianping Fan, and Xinbo Gao. “Hierarchical convolutional neural network via hierarchical cluster validity based visual tree learning”. In: *Neurocomputing* 409 (2020), pp. 408–419.