Vers un apprentissage sans exemple plus réaliste

Yannick Le Cacheux

le cnam

ÉCOLE DOCTORALE Informatique, Télécommunications et Électronique de Paris
Centre d'Études et de Recherche en Informatique et Communications

THÈSE DE DOCTORAT

présentée par : **Yannick LE CACHEUX**
soutenue le : **10 décembre 2020**

pour obtenir le grade de : **Docteur du Conservatoire National des Arts et Métiers**

Discipline : **Informatique**

Spécialité : **Apprentissage automatique**

Toward more practical zero-shot learning

THÈSE dirigée par
M. Crucianu Michel — *Professeur des Universités, CNAM*

et co-encadrée par
M. Le Borgne Hervé — *Chargé de recherche, CEA*

RAPPORTEURS
M. Jurie Frédéric — *Professeur des Universités, Université de Caen-Normandie*
M. Gravier Guillaume — *Directeur de recherche, CNRS*

PRÉSIDENT DU JURY
Mme Hudelot Céline — *Professeur des Universités, CentraleSupélec*

EXAMINATEURS
M. Sahbi Hichem — *Professeur des Universités, UMPC*

Remerciements

Cette thèse a été effectuée au Laboratoire d'Analyse Sémantique Texte Image (LASTI) du Commissariat à l'Énergie Atomique (CEA), membre de l'Université Paris-Saclay, ainsi qu'au Centre d'Études et de Recherche en Informatique et Communications (CEDRIC) du Conservatoire National des Arts et Métiers (CNAM).

Je tiens tout d'abord à remercier mes deux encadrants de thèse, Michel Crucianu et Hervé Le Borgne, pour leur patience, leur gentillesse et leur disponibilité tout au long de la thèse. Ce fut un plaisir de travailler avec eux.

Je remercie Frédéric Jurie et Guillaume Gravier pour avoir accepté de rapporter mon manuscrit. Je remercie également les autres examinateurs Hichem Sahbi et Céline Hudelot, qui m'ont fait l'honneur de participer à mon jury.

Je remercie tous mes collègues et amis du laboratoire, pour les échanges fructueux et discussions passionnantes que j'ai eu la chance d'avoir avec eux.

Je remercie Coline pour sa présence et son soutien durant toute la durée de ma thèse.

Je remercie mes colocataires Raphaël et Adrien pour les moments de détente nécessaires pour faire diminuer la pression. Je remercie également tous les amis sur qui j'ai eu la chance de pouvoir compter pendant cette période, et dont la liste exhaustive serait bien trop longue.

Enfin, je remercie mes parents, mon frère et ma soeur, ma grand-mère et mon grand-père, ainsi que l'ensemble de ma famille pour leur soutien et leur amour durant la thèse et toutes les années qui ont précédées.

Résumé

Cette thèse porte sur la reconnaissance visuelle « zero-shot », qui vise à classifier des images de catégories non rencontrées par le modèle pendant la phase d'apprentissage. Après avoir classé les méthodes existantes en trois grandes catégories, nous défendons l'idée que les méthodes dites de classement se basent habituellement sur plusieurs hypothèses implicites préjudiciables. Nous proposons d'adapter leur fonction de coût pour leur permettre d'intégrer des relations inter et intra-classe. Nous proposons également un processus permettant de diminuer l'écart entre les performances sur les classes vues et non vues dont souffrent fréquemment ces méthodes. Dans notre évaluation expérimentale, ces contributions permettent à notre modèle d'égaler ou surpasser les performances des méthodes génératives, tant en étant moins restrictif. Dans un second temps, nous nous intéressons aux représentations sémantiques utilisées dans un contexte d'application à grande échelle. Dans ce contexte, l'information sémantique provient généralement de plongements lexicaux des noms de classe. Nous soutenons que les plongements habituels souffrent d'un manque de contenu visuel dans les corpus servant à leur apprentissage. Nous proposons donc de nouveaux corpus de texte davantage connotés visuellement, ainsi qu'une méthode permettant d'adapter les modèles de plongement à ces corpus. Nous proposons en outre de compléter ces représentations non supervisées par de courtes descriptions en langage naturel, dont la production ne requiert qu'un effort minimal comparé à des attributs génériques.

Mots-clés : apprentissage zero-shot, reconnaissance visuelle, apprentissage automatique

Abstract

This thesis focuses on zero-shot visual recognition, which aims to recognize images from unseen categories, i.e. categories not seen by the model during training. After categorizing existing methods into three main families, we argue that ranking methods habitually make several detrimental implicit assumptions. We propose to adapt the usual formulation of the hinge rank loss so that such methods may take inter and intra-class relations into account. We also propose a simple process to address the gap between accuracies on seen and unseen classes, from which these methods frequently suffer in a generalized zero-shot learning setting. In our experimental evaluation, the combination of these contributions enables our proposed model to equal or surpass the performance of generative methods, while being arguably less restrictive. In a second part, we focus on the semantic representations used in a large-scale zero-shot learning setting. In this setting, semantic information customarily comes from word embeddings of the class names. We argue that usual embeddings suffer from a lack of visual content in training corpora. We thus propose new visually oriented text corpora as well as a method to adapt word embedding models to these corpora. We further propose to complete unsupervised representations with short descriptions in natural language, whose generation requires minimal effort when compared to extensive attributes.

Keywords: zero-shot learning, image recognition, machine learning.

Contents

List of Tables

List of Figures

Introduction

Computer vision is becoming of increasing importance in many scientific and industrial fields. Handwritten digits on bank checks or postal mail have been read and processed automatically for years [84, 85]. Smartphones can now be effortlessly unlocked using face recognition [134, 133]. Crop yields can be monitored from aerial and satellite images [106]. Early detection of cancer cells in medical images may soon contribute to save thousands of lives [58]. Autonomous driving promises to revolutionize mobility [27]. And many future game-changing innovations are probably just waiting to be envisioned.

Yet, giving visual abilities to a computer is not straightforward. Even something as simple for a human as differentiating pictures of cats and dogs is not straightforward to computerize. Defining precisely on a pixel-level the properties that an image must possess to represent a cat or any other high-level category proved to be an insurmountable challenge. Instead, computer vision practitioners use a completely different paradigm: rather than attempting to explicitly specify the defining features of a cat as fixed rules, they let the program learn the differences between cats and dogs from examples. This generic approach is widely known as *machine learning*. More specifically, virtually all the applications mentioned above rely on *deep convolutional neural networks* [86]. These architectures produce higher- and higher-level features, computed sequentially using kernels whose parameters are *learned* by the model on a large number of training examples. As an example, AlexNet [72], the architecture that won the 2012 edition of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [132] and arguably initiated the latest deep learning revolution [83], had access to more than one million labeled images. While seemingly impressive, this amount of training data is now dwarfed by recent models [143, 16] trained on datasets with 300 million images [143].

Despite allowing unmatched performance, deep learning models' need for data uncovers other challenges. The most obvious one is the important human annotation effort required to provide the large amount of labels necessary to train a deep model in a supervised learning setting. This may limit applications when stakeholders do not have the resources or means necessary to implement such a large investment. Furthermore, there are some classes for which it may be hard to collect hundreds or thousands of images. As an example, the saola, a critically endangered, antelope-like species from Vietnam, has only been photographed a handful of times in the wild since its discovery in 1992 [147]. More critically, the outputs of deep learning models are integrally dependent on the data used to train the model. This raises additional concerns regarding the quality and objectivity of data, as human

biases reflected in training datasets may have dire consequences [127].

Active efforts have been exerted by the research community to address this insatiable need for data. The *transfer learning* framework aims to transfer "knowledge" acquired by a model on a source problem onto a target problem, and thus reduce the amount of data necessary for the target problem. The main idea can be illustrated by a model trained to classify cats and dogs, and then retrained to classify tigers and wolves: its ability to identify snouts and fur patterns acquired from the source problem may also be useful in the target problem [145]. In practice in the case of convolutional networks, this often consists in re-using the weights of low-level kernels [125, 25]. A bit more specifically, the task of *few-shot learning* consists in designing models capable of accurately recognizing new categories after being exposed to only a few training examples, usually by heavily re-using abilities previously acquired on source problems [99, 137]. *One-shot learning* is the extreme application of this idea, where only a single training sample is allowed in order to assimilate new categories [39, 74].

And yet, the task of *zero-shot learning* aims to bring this strategy one step further. The goal of this ultimate exercise in terms of data frugality is to design models capable of recognizing objects from categories for which *no* training examples are provided [77, 75, 112]. The basic principle can be illustrated by the human ability to relate visual and non visual contents. For instance, someone who has never seen a single picture or illustration of a tiger – and naturally has never seen one in real life either – should still be able to recognize one instantly if they were told that a tiger is similar to a (very) big orange cat with black stripes and a white belly. Quite evidently, there needs to be some sort of semantic information similar to the "striped orange cat" description regarding the class *tiger* for zero-shot recognition to be possible. In this sense, zero-shot basic principles are actually quite different from few- and one-shot learning, as these tasks are usually purely visual tasks. On the other hand, zero-shot learning is by essence a *multimodal* task, which requires the ability to relate content from the visual modality (i.e. images) to at least one non visual modality (e.g. text, attributes...). More precisely, in this document, we consider that the term *zero-shot learning* refers to the design and training of a model whose goal is to classify images from *unseen* classes, for which no training examples are provided. Instead, training instances from strictly different *seen* classes can be accessed by the model during training. In addition, *semantic information* is provided for both seen and unseen classes.

Historically, this concept of zero-shot learning emerged more than a decade ago, with pioneering works such as the ones from Larochelle *et al.* [77], who first performed classification on test classes distinct from training classes, or Lampert *et al.* [75], who used attributes such as *"black"*, *"brown"* or *"has stripes"* to classify images of animal species for which no training examples were available. Around the same time, Farhadi *et al.* [37] likewise emphasized the relevance of predicting attributes from images in order to relate them to attributes from different classes; Palatucci *et al.* [112] similarly attempted to classify "unseen" words from functional magnetic resonance images (fMRI) of neural activity, using semantic representations constructed from either attributes or word co-occurrence statistics. These pioneering methods were generally fairly simple, but nevertheless led to promising results for this novel and challenging task.

This quickly sparked interest in the computer vision community, and new models and benchmark datasets were quickly introduced [129, 99, 2, 40, 138, 108]. Different settings were considered: Socher *et al.* [138] introduced a novelty detection mechanism, so that models could recognize both unseen *and* seen classes, a setting which later became known as *generalized* zero-shot learning [24]. Rohrbach *et al.* [128] popularized the *transductive* zero-shot learning task, in which unlabeled samples from unseen classes are available during training [42, 69]. Multi-label zero-shot recognition [98, 44], zero-shot detection [8, 124, 32] or zero-shot segmentation [19] were also proposed. Zero-shot learning was conjointly applied to other modalities, such as video and action recognition [93, 52] or natural language understanding [13, 141]. In parallel, the "deep learning revolution" gained momentum in computer vision, with better performing architectures being regularly introduced [136, 144, 55]. The use of these pre-trained networks as feature extractors [125, 25, 120] enabled zero-shot learning to benefit from these progresses [40, 108].

As a general framework aiming to drastically reduce the amount of data required to train models, zero-shot learning is arguably all the more relevant in a large scale setting. Consequently, Rohrbach *et al.* [129] proposed to employ 200 out of 1000 classes from ILSVRC as unseen test classes, making use of hierarchical information from WordNet to create class representations. Frome *et al.* [40] pushed the scale even further, by using the 1000 classes from ILSVRC as seen training classes and 20,000 additional classes from ImageNet as unseen test classes. As providing attributes for thousands of classes is impractical, scalable semantic representations are required in such a large scale setting. These representations usually take the form of *word embeddings* [100, 102], which consist in rich

vector representations of words capturing interesting semantic properties. The embeddings have the big advantage of typically coming from models trained on huge text corpora in an unsupervised manner, and thus of requiring close to no human annotation effort. It was thus proposed to use pre-trained word embeddings of class names as semantic representations in a large scale zero-shot learning setting [40, 138, 108].

The ability to successfully classify images in this setting could arguably be considered as the "Holy Grail" of effort-efficient approaches, as this could theoretically produce models capable of recognizing thousands of classes with close to no human annotation effort. However, in practice, performance remains modest, with reported accuracies on standard large scale benchmarks arguably too low for many practical use cases [53]. In general, performance of zero-shot learning models is unsurprisingly lower than performance of standard supervised models [164]. In addition, most zero-shot learning approaches tend to suffer from additional limitations. For instance, in the more realistic generalized zero-shot learning setting in which test classes can be either seen or unseen, many existing models tend to predict seen classes far more often than unseen classes [24, 163], which greatly decreases performance on the latter and thus the interest of using zero-shot recognition. This imbalance between seen and unseen classes is partly reduced with recent generative approaches [17, 152, 162], but this comes at the cost of more restrictive hypotheses, since contrary to other approaches, the addition of new classes often requires additional training for such models.

In this thesis, we attempt to address some of these limitations to efficient large scale zero-shot learning. We analyse existing approaches to zero-shot learning, and in particular models based on the *hinge rank loss*. We argue that previous models of this family implicitly make several assumptions regarding the nature of classes and training samples, and that these assumptions may not be justified in practice. In particular, these models typically consider that all classes are "equally different", meaning that no two classes are considered closer to each other than to other classes. Previous models further assume that every training instance is representative of its corresponding class. On the contrary, we argue that this is not the case in practice, and that failing to account for these two factors may be detrimental to the performance of the model. We thus propose a model capable of taking these elements into account, with the aim of improving the robustness of the learned multi-modal relations.

We also consider the performance gap between seen and unseen classes in a generalized zero-shot

learning setting. We investigate theoretical aspects of this phenomenon, and propose a simple process to reduce the difference in accuracy between instances from seen and unseen classes. Experiments are conducted to test the effectiveness of this process as well as the performance of the previously proposed model. Results confirm that the combination of both propositions enable to obtain state-of-the-art results in the tasks of classical and generalized zero-shot recognition. The proposed approach also has the advantage of enabling the effortless addition of new unseen classes to a trained model: contrary to most existing generative approaches, no additional training is required.

Keeping in mind the objective of keeping annotation efforts to a minimum, we investigate the role of semantic representations obtained in an unsupervised manner which are typically employed in a large scale setting, as this aspect is surprisingly under-studied in the current literature. We argue that generic text corpora may not be suitable to generate embeddings capturing meaningful visual properties of words, and instead propose new corpora together with a suitable pre-processing method. We conduct extensive experiments to measure the impact of this approach and explore its limitations.

Nonetheless, in spite of significantly improved results enabled by our proposed method, we argue that using word embeddings of class names as semantic representations may eventually have insurmountable limitations. We thus propose a compromise between employing unsupervised embeddings requiring absolutely no effort and laboriously providing extensive attributes, in the form of using short sentence descriptions in natural language. We propose several approaches to exploit such sentences, and eventually opt for semantic representations consisting of combinations of unsupervised representations and short sentence descriptions. We show that this combination enables to obtain state-of-the-art results in a large scale zero-shot learning setting, while keeping the amount of human annotation effort required at a fairly reasonable level.

This manuscript is organized as follows:

- In Chapter 1, we provide a generic overview of the field of zero-shot learning for visual recognition. In particular, we introduce the different existing settings such as the generalized or transductive settings; we present the main families of approaches; and we explain how visual features and semantic representations are usually obtained. We also provide more details on tasks relevant to this document such as generalized zero-shot learning.

- In Chapter 2, we focus on identifying unjustified assumptions made by existing models and hinge rank loss models in particular, and we introduce a model taking the corresponding aspects into account. In addition, we attempt to address the gap between seen and unseen classes mentioned previously by providing theoretical insight and corresponding empirical evidence, and we propose a simple process to address this gap. We provide detailed experiments to evaluate the impact of the proposed approaches.

- In Chapter 3, we focus more specifically on the semantic representations used for large scale zero-shot learning. We collect new text corpora arguably more suitable for the creation of visually discriminative embeddings, and propose a process to train embeddings from these corpora. We also introduce several approaches to employ short sentence descriptions as semantic embeddings. We provide experimental results for these different methods.

- Finally, Chapter 3.6 provides a summary of these contributions as well as directions for future research.

At least some contributions from most parts of this document were published in different scientific venues. The publications corresponding to each part are provided in Chapter 3.6. Furthermore, the code corresponding to the experiments has most of the time been made publicly available. This information is also provided in Chapter 3.6.

Throughout this document with maybe the exception of this introduction and Section 1.1.1, we assume that the reader has working knowledge of machine learning and deep learning. In particular, we assume the reader is familiar with the – non exhaustive – concepts of a loss function, overfitting, regularization, gradient descent, back-propagation, convolutional neural networks... We refer the reader to [12] for an introduction to machine learning and to [46] for an introduction to deep learning if needed.

Chapter 1

State-of-the-Art

Content

In this chapter, we provide a generic overview of the field of zero-shot learning, and particularly of the different experimental settings, the different families of methods, and the visual and semantic representations frequently employed. Importantly, although the term *zero-shot learning* may have several meanings, we consider in this chapter and in virtually all of this document that this refers to the task of designing a training a model whose end goal is the classification of images from classes not seen during training. This is in contrast to tasks where the end goal may be image segmentation [19], or where the entities to classify may be sentences [141], which are beyond the scope of this document.

Other surveys of the field of zero-shot learning [43] or categorizations of zero-shot learning methods have been produced [161], including fairly recently [157]. However, the organisation of this section and

of Section 1.2 in particular is fairly different, as we categorize methods by the type of loss function they employ. We also provide significantly more details on many approaches. As a result, we do not attempt to be fully comprehensive. As of October 2020, a query with the exact wording "zero-shot learning" in Google Scholar returns more than 5000 results, so we argue that such an endeavour is not reasonably feasible anyway.

This chapter is organized as follows: Section 1.1 provides a general public introduction to zero-shot learning (Section 1.1.1), introduces a more formal framework with a more detailed description of a simple method (sections 1.1.2 and 1.1.3), and introduces different settings such as transductive or generalized zero-shot learning. Section 1.2 provides a broad categorization of zero-shot learning models organized as baseline models (Section 1.2.1), least square regression models (Section 1.2.2), hinge rank loss models (Section 1.2.3) and generative models (Section 1.2.4). Specific details are included for several noteworthy models from each category. Section 1.3 provides information on the main types of visual features and semantic representations most frequently employed by zero-shot learning methods. Finally, Section 1.4 provides more details on the generalized zero-shot learning setting, as such details will be necessary in other chapters.

1.1 An introduction to zero-shot learning

1.1.1 What is zero-shot recognition?

Figure 1.1 – A giraffe and a tiger, not necessarily in this order.

The *zero-shot recognition* task consists in assigning the correct names to items that the entity performing the classification has never encountered before. Most humans are reasonably capable of this feat.

As an example, let's imagine a person that has never seen a tiger in his life, neither directly nor through any indirect visual support such as books, movies or cartoons. Let's further imagine that this person has never seen or heard of a giraffe. We give this person the following descriptions, adapted from their respective Wikipedia entries:

> *The tiger is the largest extant cat species. It is most recognisable for its dark vertical stripes on orange-brown fur with a lighter underside.*

> *The giraffe's chief distinguishing characteristics are its extremely long neck and legs, its horn-like protuberances, and its distinctive brown patches on a lighter coat.*

We then proceed to show them pictures of tigers and giraffes as in Figure 1.1 and ask them to tell us which ones correspond to a tiger and which ones correspond to a giraffe. Provided this person speaks English, has previously seen other animals such as cats, understands the concepts of stripes, necks, colors and fur, is of average intelligence and is willing to cooperate, it is reasonable to expect that they will successfully be able to complete this exercise.

Notation	Meaning
x	a scalar
$\mathbf{x}$	a vector
$\mathbf{X}$	a matrix
x_i or $(\mathbf{x})_i$	the i^{th} element (scalar) of vector $\mathbf{x}$
$X_{i,j}$ or $(\mathbf{X})_{i,j}$	the element (scalar) at line i and column j of matrix $\mathbf{X}$
$\mathbf{I}_D$	the $D \times D$ identity matrix
$\mathbf{0}_D$ / $\mathbf{1}_D$	the D-dimensional vector whose elements are all 0 / 1
$\text{diag}(\mathbf{a})$	the square diagonal matrix whose diagonal elements are the elements of vector $\mathbf{a}$
$\mathbf{a}^2$	the vector whose elements are the elements of $\mathbf{a}$ squared
$\lvert\mathbf{A}\rvert$	the determinant of $\mathbf{A}$
$\mathbf{A}^{-1}$	the inverse of $\mathbf{A}$
$\mathbf{A}^\top$	the transpose matrix of $\mathbf{A}$
$f(\cdot)$	a function returning a scalar
$\mathbf{f}(\cdot)$	a function returning a vector
$f(\cdot\,;\mathbf{w})$ or $f_\mathbf{w}(\cdot)$	a function parameterized by $\mathbf{w}$
$F[\cdot]$	a functionnal (taking a function as input and returning a scalar)
$\lVert\cdot\rVert_p$	the p-norm
$\lVert\cdot\rVert_2$ or $\lVert\cdot\rVert$	the euclidean norm for vectors or the Frobenius norm for matrices
$\odot$	the Hadamard (element-wise) product
$f(\mathbf{a})$ / $f(\mathbf{A})$	the element-wise application of f on the elements of $\mathbf{a}$ / $\mathbf{A}$
$\mathbb{1}[\cdot]$	the indicator function (1 if $\cdot$ is true, 0 otherwise)
$P(\cdot)$	a discrete probability
$p(\cdot)$	a continuous probability density function
$\mathcal{N}(\cdot\lvert\boldsymbol{\mu}, \boldsymbol{\Sigma})$	the multivariate gaussian density function
$\{x_k\}_{k\in[\![1,K]\!]}$	a set of K elements $x_1 \dots x_K$
$\lvert\{x_k\}_k\rvert$	the cardinal of a set
$\exp(\cdot)$	the exponential function
$\log(\cdot)$	the natural logarithm function
$\sigma(\cdot)$	the sigmoid function
$\tanh(\cdot)$	the hyperbolic tangent function
$[\,\cdot\,]_+$	the function $\max(\,\cdot\,, 0)$

Table 1.1 – Mathematical notations

Symbol	Meaning
$\mathcal{C}^{\mathcal{S}}$	the set of seen classes
$\mathcal{C}^{\mathcal{U}}$	the set of unseen classes
$\mathcal{C}$	the set of all classes $\mathcal{C}^{\mathcal{S}} \cup \mathcal{C}^{\mathcal{U}}$
$\mathcal{V}$	the visual space $\mathbb{R}^D$
$\mathcal{S}$	the semantic space $\mathbb{R}^K$
$\mathbf{x}_n \in \mathbb{R}^D$	the D-dimensional visual features of the n^{th} image
$y_n \in \mathcal{C}$	the label of the n^{th} image
$\mathbf{s}_c \in \mathbb{R}^K$	the K-dimensional semantic prototype of class c
$\mathbf{t}_n \in \mathbb{R}^K$	the prototype $\mathbf{s}_{y_n}$ associated with $(\mathbf{x}_n, y_n)$
$\mathbf{X} \in \mathbb{R}^{N \times D}$	the visual features of all N images, arranged in lines
$\mathbf{y} \in \mathcal{C}^N$	the labels associated with all N images
$\mathbf{S} \in \mathbb{R}^{C \times K}$	the semantic prototypes of all C classes, arranged in lines
$\mathbf{T} \in \mathbb{R}^{N \times K}$	the semantic prototypes corresponding to individual images
$\mathbf{x}^{\text{tr}}$ / $\mathbf{x}^{\text{te}}$	a training / testing visual instance
$\mathbf{x}_m^c$	the m^{th} instance of class c
$\mathcal{D}^{\text{tr}}$	the training dataset $(\{(\mathbf{x}_n, y_n)\}_{n \in [\![1,N]\!]}, \{\mathbf{s}_c\}_{c \in \mathcal{C}^{\mathcal{S}}})$
$\mathcal{D}^{te}$	the testing dataset
$f(\mathbf{x}, \mathbf{s}; \mathbf{w})$	a compatibility function $\mathbb{R}^D \times \mathbb{R}^K \to \mathbb{R}$, with parameter $\mathbf{w}$
$\mathcal{L}(\,\cdot\,)$	a loss function
$\lambda\Omega[f]$	a regularization on f, weighted by hyper-parameter λ

Table 1.2 – Frequently used notations

Abbreviation	Meaning
ZSL	Zero-Shot Learning
GZSL	Generalized Zero-Shot Learning
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
VAE	Variational Auto-Encoder
SVM	Support Vector Machine
SGD	Stochastic Gradient Descent

Table 1.3 – Frequently used abbreviations

We can identify three main ingredients which are necessary to achieve this task:

- Things to recognize, in our case tigers and giraffes.
- Semantic information on things to recognize. For instance, the fact that tigers have stripes, giraffes have a long neck, etc.
- The ability to link the semantic information to the things to classify. This generally requires previous experience involving both visual instances and the semantic information. For example, the concept of striped fur can be learned by seeing zebras and being told the alternating black and white patterns on its back are called "stripes".

Learning to associate the visual and semantic features with the objective to perform zero-shot recognition is usually called *Zero-Shot Learning* (abbreviated ZSL), although zero-shot learning and zero-shot recognition are often used interchangeably.

1.1.2 A simple example

Task and notations. We start by describing a very simple way to train a model to perform zero-shot recognition and introduce some useful notations in the process. Such notations are summarized in Table 1.2. In addition, mathematical notations used in this document are similarly summarized in Table 1.1, while frequently used abbreviations such as ZSL are summarized in Table 1.3.

The objective is to train a model to classify images belonging to categories that the model has never seen before. Such categories will be referred to as *unseen classes*, as opposed to *seen classes*, which can – and should – be used to train the model. In the previous example, *tiger* and *giraffe* would be considered unseen classes, whereas *cat* and for example *zebra* would be seen classes. Seen and unseen classes are sometimes also called respectively *source* and *target* classes. We will call the set of seen classes $\mathcal{C}^{\mathcal{S}}$, and the set of unseen classes $\mathcal{C}^{\mathcal{U}}$. The set of all classes is $\mathcal{C} = \mathcal{C}^{\mathcal{S}} \cup \mathcal{C}^{\mathcal{U}}$, with $\mathcal{C}^{\mathcal{S}} \cap \mathcal{C}^{\mathcal{U}} = \emptyset$.

For each class $c \in \mathcal{C}$, *semantic information* is provided. For now, let us suppose that this information consists of binary attributes, for example *"is orange"*, *"has stripes"*, *"has hooves"* and *"has a long neck"*. In such a case, the semantic representation for *tiger* is the vector $(1\ 1\ 0\ 0)^\top$ since a tiger is – at

least partly – orange, has stripes, and does not have hooves nor a long neck[1]. Similarly, the semantic representation for *giraffe* can be $(1\ 0\ 1\ 1)^\top$. For a given class c, we call $\mathbf{s}_c$ its corresponding semantic representation vector. Such a vector is also called the *class prototype*. Prototypes of all classes have the same dimension, which we call K, and represent the same attributes. More generally, the semantic information does not need to be binary attributes, and may not need to be attributes at all. More details on the most common types of prototypes are provided later in Section 1.3.2. But for now, suffice is to say that most ZSL methods can be applied as long as all classes are represented by vectors with the same characteristics (number of dimensions and meaning of each dimension).

Additionally, for each class c, a set of images $\{\mathcal{I}_1, \ldots, \mathcal{I}_M\}$ is available. Raw images are not always convenient to work with, particularly in a ZSL context. For this reason, we don't usually make use of the raw pixels themselves, but rather of a vector of high-level visual features extracted from this image. For image $\mathcal{I}_i$, we write this vector $\mathbf{x}_i \in \mathbb{R}^D$, D being the fixed dimension of the visual features space. $\mathbf{x}_i$ is typically extracted from $\mathcal{I}_i$ using a pre-trained deep neural network; more details on this process are provided in Section 1.3.1. For the sake of brevity, we will sometimes refer to $\mathbf{x}_i$ as the i^{th} image, although it is not strictly speaking an image. We will also refer to $\mathbf{x}_i$ as a visual sample, or an instance of class c.

Zero-shot recognition is customarily achieved in two phases: a *training* and a *testing* phase. During the training phase, the model typically only has access to the semantic representations of seen classes $\{\mathbf{s}_c\}_{c\in\mathcal{C}^\mathcal{S}}$, and to the images (or corresponding visual features) associated with these classes. We usually write $\{\mathbf{x}_n\}_{n\in[\![1,N]\!]}$ the set of N images available to the model during the training phase. $y_n \in \mathcal{C}^\mathcal{S}$ refers to the class associated with, or *label*, of training image $\mathbf{x}_n$. The full training dataset, i.e. the information to which the model has access during the training phase, is thus $\mathcal{D}^{\text{tr}} = (\{(\mathbf{x}_n, y_n)\}_{n\in[\![1,N]\!]}, \{\mathbf{s}_c\}_{c\in\mathcal{C}^\mathcal{S}})$.

During the testing phase, the model has access to the semantic representations of unseen classes $\{\mathbf{s}_{c'}\}_{c'\in\mathcal{C}^\mathcal{U}}$, and to the N' unlabeled images belonging to unseen classes $\{\mathbf{x}_{n'}\}_{n'\in[\![1,N']\!]}$. We will sometimes write $\mathbf{x}^{\text{tr}}$ or $\mathbf{x}^{\text{te}}$ to make it explicit whether $\mathbf{x}$ belongs to the training (tr) or testing (te) set, if there is a possible ambiguity. The objective for the model is to make a prediction $\hat{y}_{n'} \in \mathcal{C}^\mathcal{U}$ for each test image $\mathbf{x}^{\text{te}}_{n'}$, assigning it to the most likely unseen class.

We now have novel things to recognize, semantic information regarding these things, and training

[1]These classes and these attributes are actually part of the Animals with Attributes dataset [76, 161], described in more details in Appendix A.1.

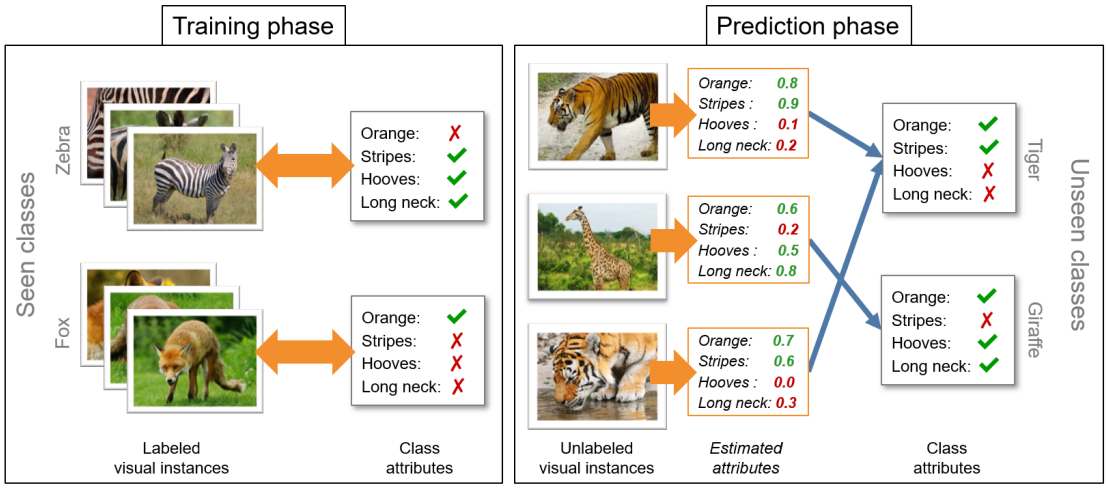

Figure 1.2 – Illustration of a simple zero-shot learning (ZSL) model. In the training phase, the model learns the relationship between visual instances and class attributes. In the prediction phase, the model estimates the presence of attributes in test images, and predicts classes from the corresponding prototypes.

data with similar semantic information associated to visual samples. All that is missing is a way to link the semantic and visual features.

A first zero-shot learning model. The following approach was first proposed in [75], and is one of the very first attempts to perform zero-shot recognition. It consists in a simple probabilistic framework, which computes the probability of presence of fixed binary attributes for a given visual input, and uses these estimated probabilities to predict the most likely unseen class corresponding to the image.

Let us recall that for each class c, we have a semantic representation $\mathbf{s}_c$ composed of binary attributes. Although the *value* of the attributes may change, the *attributes* themselves are the same for all classes. Let's call a_1 the first of these attributes, for example *"is orange"*. Similarly, we will call the other attributes $a_2, \ldots, a_K$. For every training image $\mathbf{x}_n$, we know its class y_n and have access to the corresponding semantic attributes $\mathbf{s}_{y_n}$. The corresponding k^{th} attribute a_k of image $\mathbf{x}_n$ is therefore $(\mathbf{s}_{y_n})_k \in \{0, 1\}$.

For any attribute a_k, we can thus build a labeled training set $\mathcal{D}_k = \{(\mathbf{x}_n, (\mathbf{s}_{y_n})_k)\}_n$ with the training images and corresponding labels indicating whether attribute a_k is associated with each image. We can then use this dataset to train a binary classifier to predict $P(a_k|\mathbf{x})$, the probability that attribute

a_k is present in an image $\mathbf{x}$. This classifier can be a simple logistic regression[2], in which case

$$P(a_k = 1|\mathbf{x}) = \sigma(\mathbf{w}_k^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}_k^\top \mathbf{x}}} \tag{1.1}$$

where $\sigma(\cdot)$ is the sigmoid function and $\mathbf{w}_k$ are the classifier's parameters.

At test time, given an unlabeled test image $\mathbf{x}$ belonging to one of the unseen classes, a few simplifying assumptions – detailed in Section 1.2.1 – lead to predict label $\hat{y}$ corresponding to the unseen class which maximizes the probability to observe the predicted attributes:

$$\hat{y} = \underset{c \in \mathcal{C}^\mathcal{U}}{\operatorname{argmax}} \prod_{k=1}^{K} P(a_k = (\mathbf{s}_c)_k|\mathbf{x}) \tag{1.2}$$

This process is illustrated in Figure 1.2.

1.1.3 Formal framework

More generally, many ZSL methods in the literature are based on a *compatibility function* $f : \mathbb{R}^D \times \mathbb{R}^K \to \mathbb{R}$ assigning a "compatibility" score $f(\mathbf{x}, \mathbf{s})$ to a pair composed of a visual sample $\mathbf{x} \in \mathbb{R}^D$ and a semantic prototype $\mathbf{s} \in \mathbb{R}^K$. Ideally, if the visual sample corresponds to the semantic description, e.g. if $\mathbf{x}$ is of class c and $\mathbf{s}$ is $\mathbf{s}_c$, $f(\mathbf{x}, \mathbf{s})$ should be high – and vice versa. This function may be parameterized by a vector $\mathbf{w}$ or a matrix $\mathbf{W}$, or by a set of parameters $\{\mathbf{w}_i\}_i$. We will write $f_\mathbf{w}(\mathbf{x}, \mathbf{s})$, $f(\mathbf{x}, \mathbf{s}; \{\mathbf{w}_i\}_i)$ or use similar notations when we need to explicitly refer to these parameters. These parameters are generally learned using the training dataset $\mathcal{D}^{\text{tr}} = (\{(\mathbf{x}_n, y_n)\}_{n \in [\![1,N]\!]}, \{\mathbf{s}_c\}_{c \in \mathcal{C}^\mathcal{S}})$ by applying a suitable loss function $\mathcal{L}$ and by minimizing the total training loss $\mathcal{L}^{\text{tr}}$ over the training dataset $\mathcal{D}^{\text{tr}}$ with respect to the parameters $\mathbf{w}$:

$$\mathcal{L}^{\text{tr}}(\mathcal{D}^{\text{tr}}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{c \in \mathcal{C}^\mathcal{S}} \mathcal{L}[(\mathbf{x}_n, y_n, \mathbf{s}_c), f_\mathbf{w}] + \lambda \Omega[f_\mathbf{w}] \tag{1.3}$$

where $\Omega[f]$ is a regularization penalty based on f and weighted by the hyper-parameter λ (or in some cases by a set of hyper-parameters), which aims to reduce overfitting and thus increase the generalization abilities of the model [12].

Obtaining the global minimum of the cost in Equation (1.3) is not always possible, and sometimes not desirable. In such cases, iterative numerical optimization algorithms may be used. Examples of

[2] In [75], either a logistic regression or a support vector machine (SVM) classifier with Platt scaling [119] is used to estimate probabilities.

such algorithms include a standard stochastic gradient descent (SGD), AdaGrad [34], RMSProp [56] or Adam [65].

After the training phase – once "good" parameters $\mathbf{w}$ have been learned – and given a test image $\mathbf{x}$, the predicted label $\hat{y}$ can be selected among candidate testing classes based on their semantic representations $\{\mathbf{s}_c\}_{c\in\mathcal{C}^u}$:

$$\hat{y} = \underset{c\in\mathcal{C}^U}{\operatorname{argmax}}\ f_{\mathbf{w}}(\mathbf{x}, \mathbf{s}_c) \tag{1.4}$$

Link with the previous example. In our previous example in Section 1.1.2, the compatibility function f between $\mathbf{x}$ and $\mathbf{s}$ was the estimated probability to observe attributes corresponding to $\mathbf{s}$ given image $\mathbf{x}$, assuming independence of the K attributes and using logistic regressions to estimate probabilities. It could be written

$$f(\mathbf{x}, \mathbf{s}; \{\mathbf{w}_k\}_k) = \prod_{k=1}^{K} \Big(s_k\ \sigma(\mathbf{w}_k^\top \mathbf{x}) + (1 - s_k)(1 - \sigma(\mathbf{w}_k^\top \mathbf{x})) \Big) \tag{1.5}$$

Each of the parameters $\mathbf{w}_k$ are learned on a training set $\mathcal{D}_k = \{(\mathbf{x}_n, (\mathbf{s}_{y_n})_k)\}_n$ using a standard log loss $\mathcal{L}_{\log}$:

$$\mathcal{L}_{\log}(\mathbf{x}, y; \mathbf{w}) = -\Big(y\ \log(\sigma(\mathbf{w}^\top \mathbf{x})) + (1 - y)\ \Big(1 - \log(\sigma(\mathbf{w}^\top \mathbf{x}))\Big)\Big) \tag{1.6}$$

$$\mathbf{w}_k = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \mathcal{L}_{\log}(\mathbf{x}_n, (\mathbf{s}_{y_n})_k; \mathbf{w}) \tag{1.7}$$

So the loss function $\mathcal{L}$ in Equation (1.3) could be written:

$$\mathcal{L}[(\mathbf{x}_n, y_n, \mathbf{s}_c), f_{\mathbf{w}}] = \begin{cases} \sum_k \mathcal{L}_{\log}(\mathbf{x_n}, (\mathbf{s}_c)_k; \mathbf{w}_k) & \text{if } \mathbf{s}_c = \mathbf{s}_{y_n} \\ 0 & \text{otherwise} \end{cases} \tag{1.8}$$

Although no explicit regularization is mentioned in [75], it would be straightforward to add an $\ell 1$ or $\ell 2$ penalty to the model:

$$\Omega_{\ell 2}[f(\cdot, \cdot\ ; \{\mathbf{w}\}_k)] = \sum_{k=1}^{K} \|\mathbf{w}_k\|_2^2 = \sum_{k=1}^{K} \sum_{i=1}^{D} \left((\mathbf{w}_k)_i\right)^2 \tag{1.9}$$

Other examples of compatibility and loss functions will be studied in more details in Section 1.2.

1.1.4 Zero-shot learning settings

So far, we considered that the only information available during the training phase was (1) the class prototypes of the *seen* classes, and (2) labeled visual samples from these seen classes. The information

Setting	Description
Information available for training	
Class-inductive	Prototypes from seen classes only
Class-transductive	Prototypes from seen and unseen classes
(Instance-)inductive	Instances from seen classes only
(Instance-)transductive	Instances from seen classes and (unlabeled) instances from unseen classes
Candidate classes during testing	
Standard supervised learning	Seen classes only
(Classical) ZSL	Unseen classes only
Generalized ZSL	Seen and unseen classes

Table 1.4 – Summary of the main zero-shot learning settings. The classical instance-inductive, class-inductive setting is assumed to be the default setting in this document.

regarding the class prototypes of *unseen* classes as well as unlabeled instances from these unseen classes was only provided during the testing phase, after the model was trained. In addition, the test samples for which we made predictions could only belong to these unseen classes; we considered they could not belong to a seen class, or to a new class for which no semantic information is available.

This corresponds to the most common ZSL setting, and the one we will assume is the default in this document. However, it is important to mention that other settings may be considered, with different information being made available at different times, or with different tasks being conducted. We will simply describe briefly the other possible settings in this section. Later sections such as Section 1.4 will be entirely dedicated to some of these settings and their specificities.

1.1.4.1 Available information at training time: inductive vs. transductive settings

Class-inductive and class-transductive settings. In some settings, class prototypes of both seen *and* unseen classes are available during the training phase. Such a setting is referred to as a *class-transductive* setting by [157], as opposed to a *class-inductive* setting in which unseen class prototypes are only made available after the training of the model is completed. In a class-transductive setting, the prototypes of unseen classes can for example be leveraged by a generative model, which tries to synthesize images of objects from unseen classes based on their semantic description (Section 1.2.4). They can also simply be used during training to ensure that the model does not misclassify a sample from a seen class as a sample from an unseen class.

Having access to this information as soon as the training phase may be legitimate for some use-

cases. However, this still implies that the model loses some flexibility: new classes cannot be added as seamlessly as in a class-inductive setting, in which a new class can be introduced by simply providing its semantic representation.

(Instance-)inductive and (instance-)transductive settings. Some settings are even more permissive, and consider that (unlabeled) instances of unseen classes are available during training. Such a setting is called an *instance-transductive* setting in [157], as opposed to an *instance-inductive* setting. These two settings are often simply referred to as respectively a *transductive* setting and an *inductive* setting, even though there is some ambiguity on whether the (instance-)inductive setting designates a class-inductive or a class-transductive setting. Some methods use approaches which specifically take advantage of the availability of these unlabeled images [41, 128], for example by extracting additional information on the geometry of the visual manifold [42, 165].

Even though models operating in and taking advantage of a transductive setting can often achieve better accuracy than models designed for an inductive setting, we can argue that such a setting is not suitable for many real-life use cases. With a few exceptions [90, 169], most transductive approaches consider that the actual (unlabeled) testing instances are available during the training phase, which excludes many practical applications. Even without this assumption, it is not always reasonable to expect that we will have access to unlabeled samples from many unseen classes during the testing phase. One may further argue that this is all the more unrealistic as there is some evidence that labeling even a single instance per class (in a "one-shot learning" scenario) can lead to a significant improvement in accuracy over a purely zero-shot learning scenario [164].

To summarize, a ZSL setting may be (instance-)inductive or (instance-)transductive depending on whether unlabeled instances of test classes are available during the training phase. An (instance-)inductive setting may further be class-inductive or class-transductive depending on whether semantic prototypes of unseen classes are available at training time, even though this aspect is not always explicitly stated in the literature and an "inductive setting" may designate both. The default class-inductive, instance-inductive setting is thus the most restrictive setting, but makes the fewest assumptions on the availability of information during the different phases and is therefore the most broadly applicable. By default, we will consider in this document that we are operating in this class-inductive,

instance-inductive setting. This will specifically be the case for models studied in Section 1.2, with a few exceptions in the section dealing with generative models (Section 1.2.4). Information on these different settings is summarized in Table 1.4.

1.1.4.2 Use of additional information

In some settings, the available information itself can be different from the default setting. For example, in addition to the semantic prototypes, some methods make use of relations between classes defined with a graph [158, 60] or a hierarchical structure [129]. Others make use of information regarding the environment of the object, for example by detecting surrounding objects [166] or by computing co-occurence statistics using an additional multilabel dataset [98]. Other methods consider that instead of a semantic representation per class, a semantic representation *per image* is available, for example in the form of text descriptions [126] or human gaze information [61]. Further details on some of these settings are provided in Section 1.3.2. Although all these settings can lead to interesting alternative problems, they are mostly out of the scope of this document.

1.1.4.3 Task during the testing phase: classical vs. generalized ZSL

So far, we considered that the test instances could only belong to one of the unseen classes, and thus only these unseen classes can be predicted. However, there is generally no particular reason to assume that the seen classes we encountered during the training phase cannot be encountered again when applying and testing our model; in a real-life use case, one may legitimately want to recognize both seen and unseen classes. The setting in which testing instances may belong to both seen and unseen classes is usually called *generalized zero-shot learning*, abbreviated GZSL.

Even though GZSL is less restrictive than standard ZSL, it presents some specific challenges which will be detailed in Section 1.4 and are not always straightforward to address. For this reason, we consider that unless stated otherwise, we are not in a GZSL setting. We will use the term *classical* ZSL as opposed to generalized ZSL when we need to make this explicit.

Other even less restrictive tasks may be considered during the testing / application phase. For instance, one may want a model able to answer that a visual instance does not match either a seen nor an unseen class. Or one may want to recognize entities belonging to several categories, a setting

known as multilabel ZSL [98, 44, 88]. However, these tasks are outside the scope of this document. Similarly, there exist transductive *and* generalized settings [140], but we dot not aim to (and are not able to) be fully comprehensive.

1.2 Standard methods

In this section, we provide an overview of the different usual types of methods which have been developed to solve the ZSL problem. Standard methods are separated into three main categories: *regression methods* (Section 1.2.2), *ranking methods* (Section 1.2.3) and *generative methods* (Section 1.2.4), in addition to really simple methods which we refer to as baselines (Section 1.2.1).

This is simply one possible categorization, and different ones have been proposed in [161, 43, 157]. As with all possible categorizations, it is sometimes not so clear which category best fits a given method; but overall, we think this choice is well-suited to give a general overview of the field. We would also like to emphasize that sometimes, methods described here are slightly different from their initial formulation in the original articles. This is done for the sake of brevity and simplicity; the aim is to give a general overview of the types of methods, their strengths, weaknesses and underlying hypotheses, not to dive deep in very specific implementation details. We usually try to disclose it explicitly when what is described is different from the cited article, and refer the reader to the original publications for further details.

Experimental results on standard ZSL benchmark datasets (presented in Appendix A.1) are available in Table 2.5 for several methods of each category, originating from the reference comparison from Xian *et al.* [161] as well as from our independent reproduction of some of these results.

1.2.1 Baselines

This section describes very simple methods, which were usually proposed early in ZSL history. These methods have the advantage of being easy to implement and requiring very little computing power – or at least, no more than regular supervised classification. Some of them even enable to directly adapt a standard supervised classifier to a ZSL setting. However, their performance is usually quite low compared to more advanced methods.

We already encountered the **Direct Attribute Prediction** or **DAP** [75] approach in Section 1.1.2. Recall that this method simply consists in training K standard classifiers which provide the probability $P(a_k|\mathbf{x})$ that attribute a_k is present in visual input $\mathbf{x}$. At test time, we predict the class c which maximizes the probability to have attributes corresponding to its class prototype $\mathbf{s}_c$ (Equation (1.2)). We now provide some additional information regarding the underlying assumptions which led to Equation (1.2).

Ideally, we would like to be able to estimate the conditional probability $P(c|\mathbf{x})$ that a visual instance $\mathbf{x}$ belong to class c. For a given test image $\mathbf{x}$, this would lead to predict label $\hat{y}$ such that

$$\hat{y} = \underset{c \in \mathcal{C}^{\mathcal{U}}}{\operatorname{argmax}} \; P(c|\mathbf{x}) \tag{1.10}$$

From the sum and product rules of probability,

$$P(c|\mathbf{x}) = \sum_{\mathbf{a} \in \{0,1\}^K} P(c|\mathbf{a}) P(\mathbf{a}|\mathbf{x}) \tag{1.11}$$

From Bayes' theorem,

$$P(c|\mathbf{a}) = \frac{P(\mathbf{a}|c)P(c)}{P(\mathbf{a})} \tag{1.12}$$

Assuming deterministic attributes $\mathbf{a} = \mathbf{s}_c$ given the class c,

$$P(\mathbf{a}|c) = \mathbb{1}[\mathbf{a} = \mathbf{s}_c] = \begin{cases} 1 & \text{if } \mathbf{a} = \mathbf{s}_c \\ 0 & \text{otherwise} \end{cases} \tag{1.13}$$

From equations (1.12) and (1.13),

$$P(c|\mathbf{a}) = \begin{cases} \frac{P(c)}{P(\mathbf{a}=\mathbf{s}_c)} & \text{if } \mathbf{a} = \mathbf{s}_c \\ 0 & \text{otherwise} \end{cases} \tag{1.14}$$

Which means we only keep one term in the sum in Equation (1.11)

$$P(c|\mathbf{x}) = P(c|\mathbf{a} = \mathbf{s}_c) P(\mathbf{a} = \mathbf{s}_c|\mathbf{x}) = \frac{P(c)}{P(\mathbf{a} = \mathbf{s}_c)} P(\mathbf{a} = \mathbf{s}_c|\mathbf{x}) \tag{1.15}$$

If we further assume identical class priors $P(c)$ and uniform attribute priors[3] $P(a_k) = \frac{1}{2}$, the only non-constant term in Equation (1.15) is $P(\mathbf{a} = \mathbf{s}_c|\mathbf{x})$. Finally, assuming independence of attributes,

$$P(\mathbf{a}|\mathbf{x}) = \prod_{k=1}^{K} P(a_k|\mathbf{x}) \tag{1.16}$$

[3]The authors of [75] also suggest to compute attribute priors $P(\mathbf{a})$ by assuming independent attributes and estimating the probability of each attribute using the training dataset, but indicate that experimental results of both approaches are similar.

which gives us the same result as in Equation (1.2):

$$\underset{c\in\mathcal{C}^{\mathcal{U}}}{\operatorname{argmax}}\ P(c|\mathbf{x}) = \underset{c\in\mathcal{C}^{\mathcal{U}}}{\operatorname{argmax}}\ \prod_{k=1}^{K} P(a_k = (\mathbf{s}_c)_k|\mathbf{x}) \tag{1.17}$$

Note that here we further assumed that the class prototypes consist of binary attributes. Similar results may be obtained with continuous attributes by using probability density functions and regressors instead of classifiers.

The **Indirect Attribute Prediction** or **IAP** was also proposed in [75]. It is very close to DAP, with one notable difference: it does not require any model training beyond a standard multi-class classifier on seen classes, and in particular does not require any training related to the attributes. As such, it enables to seamlessly convert any pre-trained standard supervised classification model to a ZSL setting provided a semantic representation is available for each seen and unseen class.

To achieve this, we can simply rewrite $P(a_k|\mathbf{x})$ in Equation (1.2) (or Equation (1.17)) as

$$P(a_k|\mathbf{x}) = \sum_{c\in\mathcal{C}^{\mathcal{S}}} P(a_k|c)P(c|\mathbf{x}) \tag{1.18}$$

Here $P(c|\mathbf{x})$ for seen class $c \in \mathcal{C}^{\mathcal{S}}$ can be obtained using any "standard" (not zero-shot learning) supervised classifier trained on the dataset $\{(\mathbf{x}_n, y_n)\}_n$, and the deterministic attributes assumption tells us that $P(a_k|c)$ is 1 if $a_k = (\mathbf{s}_c)_k$ and 0 otherwise, so

$$P(a_k|\mathbf{x}) = \sum_{\substack{c\in\mathcal{C}^{\mathcal{S}} \\ (\mathbf{s}_c)_k = a_k}} P(c|\mathbf{x}) \tag{1.19}$$

And Equation (1.2) can be rewritten as

$$\hat{y} = \underset{c\in\mathcal{C}^{\mathcal{U}}}{\operatorname{argmax}} \prod_{k=1}^{K} \sum_{\substack{c'\in\mathcal{C}^{\mathcal{S}} \\ (\mathbf{s}_c)_k = (\mathbf{s}_{c'})_k}} P(c'|\mathbf{x}) \tag{1.20}$$

From now on, we consider that values in semantic representations can be either binary or continuous, and are not restricted to $[0, 1]$. These representations may also not be attributes, as detailed in Section 1.3.2. We only assume that all prototypes have the same dimension K.

Convex semantic embeddings. The method based on convex semantic embeddings, or **ConSE** [108], is similar to IAP in that it also relies only on standard classifiers and can be used to adapt pre-trained standard models to a ZSL setting without any further training.

We assume that we already have a classifier which gives $P(c|\mathbf{x})$, the probability that visual instance $\mathbf{x}$ belongs to class c for any seen class $c \in \mathcal{C}^{\mathcal{S}}$. Such a classifier can be trivially trained on the dataset $\{(\mathbf{x}_n, y_n)\}_n$. We will write $\hat{c}_1(\mathbf{x}) = \underset{c \in \mathcal{C}^{\mathcal{S}}}{\text{argmax}}\ P(c|\mathbf{x})$ the prediction with the highest probability for input image $\mathbf{x}$. Similarly $\hat{c}_2(\mathbf{x})$ is the prediction with the second highest probability, and $\hat{c}_t(\mathbf{x})$ is the t^{th} best prediction.

Given a visual sample $\mathbf{x}$, we estimate its semantic representation $\hat{\mathbf{s}}(\mathbf{x}) \in \mathbb{R}^K$ as a convex combination[4] of the semantic prototypes $\mathbf{s}_{\hat{c}}$ of the best predictions $\hat{c}_t(\mathbf{x})$ for $\mathbf{x}$, each prototype being weighted by its classification score $P(\hat{c}_t(\mathbf{x})|\mathbf{x})$:

$$\hat{\mathbf{s}}(\mathbf{x}) = \sum_{t=1}^{T} P(\hat{c}_t(\mathbf{x})|\mathbf{x})\ \mathbf{s}_{\hat{c}_t(\mathbf{x})} \tag{1.21}$$

$\mathbf{s}_{\hat{c}_t(\mathbf{x})}$ being the class prototype associated with the t^{th} predicted seen class $\hat{c}_t(\mathbf{x})$. T is a hyperparameter of the model.

For a test instance $\mathbf{x}$, we can then simply predict $\hat{y}$ as the class c whose class prototype $\mathbf{s}_c$ is the closest to the estimated semantic representation $\hat{\mathbf{s}}(\mathbf{x})$ as measured with cosine similarity:

$$\hat{y} = \underset{c \in \mathcal{C}^{\mathcal{U}}}{\text{argmax}}\ \cos(\hat{\mathbf{s}}(\mathbf{x}), \mathbf{s}_c) = \underset{c \in \mathcal{C}^{\mathcal{U}}}{\text{argmax}}\ \frac{\hat{\mathbf{s}}(\mathbf{x})^\top \mathbf{s}_c}{\|\hat{\mathbf{s}}(\mathbf{x})\|_2 \|\mathbf{s}_c\|_2} \tag{1.22}$$

We can notice that contrary to DAP and IAP, ConSE does not make any implicit assumption regarding the nature of the class prototypes – it can be used with either binary or continuous semantic representations. It is also interesting to note that if $T = 1$, this method is equivalent to simply determining the best matching seen class to (unseen) test instance $\mathbf{x}$, and predicting the unseen class whose prototype is closest to the prototype of the best matching seen class.

1.2.2 Ridge regression

Zero-shot learning as a regression problem. One simple approach to ZSL is to view this task as a regression problem, where we are trying to predict continuous attributes from a visual instance. A very straight-forward implementation of this idea consists in learning to predict the attributes from the visual samples with a simple linear regression.

[4] To truly be a convex combination, $\hat{\mathbf{s}}(\mathbf{x})$ in Equation (1.21) should be divided by a normalization term $\sum_{t=1}^{T} P(\hat{c}_t(\mathbf{x})|\mathbf{x})$, but since we use cosine similarity to make predictions in Equation (1.22), the norm of $\hat{\mathbf{s}}(\mathbf{x})$ does not matter.

Given a visual sample $\mathbf{x}$ and the corresponding semantic representation $\mathbf{s}$, we aim to predict each semantic dimension s_k of $\mathbf{s}$ with $\hat{s}_k = \mathbf{w}_k^\top \mathbf{x}$ so as to minimize the squared distance between the prediction and the true value $(\hat{s}_k - s_k)^2$, $\mathbf{w}_k \in \mathbb{R}^D$ being the parameters of the model. If we write $\mathbf{W} = (\mathbf{w}_1, \dots, \mathbf{w}_K)^\top \in \mathbb{R}^{K \times D}$, we can directly estimate the entire prototype with $\hat{\mathbf{s}} = \mathbf{W}\mathbf{x}$. We can also directly compare how close $\hat{\mathbf{s}}$ is to $\mathbf{s}$ with $\|\hat{\mathbf{s}} - \mathbf{s}\|_2^2 = \sum_k (\hat{s}_k - s_k)^2$. As with a standard linear regression, we determine the optimal parameters $\mathbf{W}^\star$ by minimizing the training error over the training dataset $\mathcal{D}^{\text{tr}}$:

$$\mathbf{W}^\star = \operatorname*{argmin}_{\mathbf{W}} \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{W}\mathbf{x}_n - \mathbf{s}_{y_n}\|_2^2 \tag{1.23}$$

We denote $\mathbf{t}_n = \mathbf{s}_{y_n}$ the prototype associated with the class of the n^{th} image $\mathbf{x}_n$. We further denote $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ the matrix whose N lines correspond to the visual features of training samples. Similarly, we write $\mathbf{T} = (\mathbf{t}_1, \dots, \mathbf{t}_N)^\top \in \mathbb{R}^{N \times K}$. This enables us to write the loss in Equation (1.23) in matrix form $\frac{1}{N}\|\mathbf{X}\mathbf{W}^\top - \mathbf{T}\|_2^2$ – to simplify notations, we denote $\|\cdot\|_2$ both the $\ell 2$ norm $\|\mathbf{a}\|_2 = \left(\sum_i a_i^2\right)^{1/2}$ when applied to a vector, and the Frobenius norm $\|\mathbf{A}\|_F = \left(\sum_i \sum_j A_{i,j}^2\right)^{1/2}$ when applied to a matrix. Additionally, we can add a regularization in the form of a $\ell 2$ penalty on the model parameters $\mathbf{W}$, weighted by a hyperparameter λ. This results in the following loss:

$$\frac{1}{N}\|\mathbf{X}\mathbf{W}^\top - \mathbf{T}\|_2^2 + \lambda \|\mathbf{W}\|_2^2 \tag{1.24}$$

A loss of the form of Equation (1.24) has the huge advantage of having a closed-form solution, which enables to directly obtain the value of the optimal parameters. We will shortly derive such a closed-form solution from Lemma 1.26, which expresses a slightly more generic relation that will enable us to obtain solutions to other similar problems in this section.

At test time, given an image $\mathbf{x}$ belonging to an unseen class, we estimate its corresponding semantic representation $\hat{\mathbf{s}} = \mathbf{W}\mathbf{x}$ and predict the class with the closest semantic prototype:

$$\hat{y} = \operatorname*{argmin}_{c \in \mathcal{C}^{\mathcal{U}}} \|\mathbf{W}\mathbf{x} - \mathbf{s}_c\|_2 \tag{1.25}$$

It is also possible to use other distances or similarity measures such as a cosine similarity during the prediction phase.

Lemma. For matrices $\mathbf{X}$, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ defined such that the following equation makes sense, we have

$$\frac{\partial \|\mathbf{AXB}+\mathbf{C}\|_2^2}{\partial \mathbf{X}} = 2\mathbf{A}^\top(\mathbf{AXB}+\mathbf{C})\mathbf{B}^\top \tag{1.26}$$

Proof: let's write $\mathbf{Z} = \mathbf{AXB}+\mathbf{C}$.

$\|\mathbf{AXB}+\mathbf{C}\|_2^2 = \|\mathbf{Z}\|_2^2 = \sum_i \sum_j (Z_{i,j})^2$, and

$$\frac{\partial \|\mathbf{Z}\|_2^2}{\partial \mathbf{X}} = \sum_i \sum_j \frac{\partial (Z_{i,j})^2}{\partial \mathbf{X}} \tag{1.27}$$

We have $Z_{i,j} = \sum_k A_{i,k}(\mathbf{XB})_{k,j} + C_{i,j}$, or

$$Z_{i,j} = \sum_k A_{i,k}\left(\sum_l X_{k,l}B_{l,j}\right) + C_{i,j} \tag{1.28}$$

In Equation (1.27), for any $X_{m,n}$, we have

$$\frac{\partial (Z_{i,j})^2}{\partial X_{m,n}} = 2Z_{i,j}\frac{\partial Z_{i,j}}{\partial X_{m,n}} \tag{1.29}$$

Using Equation (1.28),

$$\frac{\partial Z_{i,j}}{\partial X_{m,n}} = A_{i,m}B_{n,j} \tag{1.30}$$

Combining equations (1.27), (1.29) and (1.30), we have

$$\frac{\partial \|\mathbf{Z}\|_2^2}{\partial X_{m,n}} = \sum_i \sum_j (2Z_{i,j}A_{i,m}B_{n,j}) = 2\sum_i (\mathbf{A}^\top)_{m,i}\left(\sum_j Z_{i,j}(\mathbf{B}^\top)_{j,n}\right)$$

and finally

$$\frac{\partial \|\mathbf{Z}\|_2^2}{\partial X_{m,n}} = 2(\mathbf{A}^\top\mathbf{ZB}^\top)_{m,n}$$

Q.E.D.

In the regularized least-square loss in Equation (1.24), noticing that $\|\mathbf{Z}\|_2^2 = \|\mathbf{Z}^\top\|_2^2$ enables us to use Lemma 1.26 to get the derivative with respect to $\mathbf{W}$:

$$\frac{2}{N}(\mathbf{WX}^\top - \mathbf{T}^\top)\mathbf{X} + 2\lambda\mathbf{W} \tag{1.31}$$

Setting the derivative from Equation (1.31) to 0 and rearranging, we get:

$$\mathbf{W}(\mathbf{X}^\top\mathbf{X} + \lambda N\mathbf{I}_D) = \mathbf{T}^\top\mathbf{X} \tag{1.32}$$

Which finally gives us the closed-form solution to Equation (1.24):

$$\mathbf{W} = \mathbf{T}^\top \mathbf{X}(\mathbf{X}^\top \mathbf{X} + \lambda N \mathbf{I}_D)^{-1} \tag{1.33}$$

We call this ridge-regression based method the $\textbf{Ridge}_{\mathcal{V}\to\mathcal{S}}$ method, as we learn a mapping from the visual ($\mathcal{V}$) to the semantic ($\mathcal{S}$) space.

The embarrassingly simple approach. The Embarrassingly Simple approach to Zero-Shot Learning proposed in [130], often abbreviated **ESZSL**, makes use of a similar idea with a few additional steps.

We again aim to learn parameters $\mathbf{W} \in \mathbb{R}^{K\times D}$ such that the linear projection $\hat{\mathbf{t}}_n = \mathbf{W}\mathbf{x}_n$ from the image $\mathbf{x}_n$ corresponds to "good" predicted semantic representations. However, instead of comparing the estimated attributes to the ground truth $\mathbf{t}_n = \mathbf{s}_{y_n}$ using a squared distance as in Equation (1.23), we measure how similar they are using a dot product $\hat{\mathbf{t}}_n^\top \mathbf{t}_n$. Ideally and somewhat arbitrarily, we would like this similarity to be close to 1. Conversely, we would like the similarity $\hat{\mathbf{t}}_n^\top \mathbf{s}_c$ between $\hat{\mathbf{t}}_n$ and the prototype $\mathbf{s}_c$ of another class $c \neq y_n$ to be close to -1.

We therefore introduce a matrix $\mathbf{Y} \in \{-1, 1\}^{N\times C}$ representing this objective: for an image $\mathbf{x}_n$ with label y_n, line n of $\mathbf{Y}$ is -1 everywhere except on column y_n where it is 1. Using the same notations as previously, $\mathbf{X}\mathbf{W}^\top \in \mathbb{R}^{N\times K}$ represents the predicted semantic representations for all N training images. We write $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_C)^\top \in \mathbb{R}^{C\times K}$, C being the number of seen classes $C = |\mathcal{C}^\mathcal{S}|$. Then, the matrix $\mathbf{X}\mathbf{W}^\top\mathbf{S}^\top \in \mathbb{R}^{N\times C}$ should be as close to $\mathbf{Y}$ as possible.

We can therefore use the following loss:

$$\frac{1}{N}\|\mathbf{X}\mathbf{W}^\top\mathbf{S}^\top - \mathbf{Y}\|_2^2 \tag{1.34}$$

In [130], the authors further propose to add regularization terms so that visual features projected on the semantic space $\mathbf{X}\mathbf{W}^\top$ have similar norms to allow for fair comparison. The same idea is applied to semantic prototypes projected on the visual space $\mathbf{W}^\top\mathbf{S}^\top$. They also add an $\ell 2$ penalty on the model parameters $\mathbf{W}$ as we did in Equation (1.24). This results in the final, regularized loss:

$$\frac{1}{N}\|\mathbf{X}\mathbf{W}^\top\mathbf{S}^\top - \mathbf{Y}\|_2^2 + \gamma\|\mathbf{W}^\top\mathbf{S}^\top\|_2^2 + \frac{\lambda}{N}\|\mathbf{X}\mathbf{W}^\top\|_2^2 + \gamma\lambda\|\mathbf{W}\|_2^2 \tag{1.35}$$

λ and γ being hyperparameters controlling the weights of the different regularization terms.

Using the same method as previously, we compute the derivative of Equation (1.35) using Lemma 1.26, set it to zero and rearrange the resulting equation to obtain the closed-form solution:

$$\mathbf{W} = (\mathbf{S}^\top\mathbf{S} + \lambda N\mathbf{I}_K)^{-1}\mathbf{S}^\top\mathbf{Y}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \gamma N\mathbf{I}_D)^{-1} \tag{1.36}$$

The prediction phase is again straightforward. Given a test image $\mathbf{x}$, we predict the unseen class $c \in \mathcal{C}^\mathcal{U}$ with the highest similarity to the predicted semantic representation $\hat{\mathbf{t}} = \mathbf{W}\mathbf{x}$ of $\mathbf{x}$:

$$\hat{y} = \underset{c\in\mathcal{C}^\mathcal{U}}{\operatorname{argmax}}\ \mathbf{s}_c^\top\mathbf{W}\mathbf{x} \tag{1.37}$$

From the perspective of the formal framework introduced in Section 1.1.3, this means that the compatibility function f between visual sample $\mathbf{x}$ and semantic representation $\mathbf{s}$ is a bilinear form $f(\mathbf{x}, \mathbf{s}) = \mathbf{s}^\top\mathbf{W}\mathbf{x}$. We will meet compatibility functions of this form again later in Section 1.2.3.

Predicting visual features: Ridge$_{\mathcal{S}\to\mathcal{V}}$. Instead of trying to predict the class prototypes $\mathbf{s}$ from the visual features $\mathbf{x}$, we can consider predicting the visual features from the class prototypes' features. This is very similar to what we did in the beginning of this section with the Ridge$_{\mathcal{V}\to\mathcal{S}}$ model: we predict each visual dimension x_d with $\hat{x}_d = \mathbf{w}_d^\top\mathbf{s}$, a linear combination of $\mathbf{s}$ parameterized by $\mathbf{w}_d \in \mathbb{R}^K$. Writing $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_D) \in \mathbb{R}^{K\times D}$, we can estimate the "average" visual representation associated with prototype $\mathbf{s}$ with $\hat{\mathbf{x}} = \mathbf{W}^\top\mathbf{s}$. We therefore call this method the **Ridge$_{\mathcal{S}\to\mathcal{V}}$** model, as we learn a mapping from the semantic ($\mathcal{S}$) space to the visual ($\mathcal{V}$) space.

The distances between the observations and our predictions are this time measured in the visual space: for each training sample $\mathbf{x}_n$, we want to minimize the distance $\|\mathbf{x}_n - \hat{\mathbf{x}}_n\|^2$ between the sample $\mathbf{x}_n$ and the predicted visual features $\hat{\mathbf{x}}_n = \mathbf{W}^\top\mathbf{s}_{y_n}$ of the corresponding semantic prototype $\mathbf{s}_{y_n}$. Using the same notations as previously, we can write the corresponding loss:

$$\frac{1}{N}\|\mathbf{X} - \mathbf{T}\mathbf{W}\|_2^2 + \lambda\|\mathbf{W}\|_2^2 \tag{1.38}$$

Using Lemma 1.26 and the same method as previously, we find the closed-form solution

$$\mathbf{W} = (\mathbf{T}^\top\mathbf{T} + \lambda N\mathbf{I}_K)^{-1}\mathbf{T}^\top\mathbf{X} \tag{1.39}$$

For test image $\mathbf{x}$, we predict label $\hat{y}$ based on a nearest-neighbor search in the visual space:

$$\hat{y} = \underset{c\in\mathcal{C}^\mathcal{U}}{\operatorname{argmin}}\ \|\mathbf{x} - \mathbf{W}^\top\mathbf{s}_c\|_2 \tag{1.40}$$

Although this approach is very similar to the first approach of this section, it turns out projecting semantic objects to the visual space has an advantage compared to doing the opposite. Like other machine learning methods, ZSL methods can be subject to the *hubness problem* [123], which describes a situation where certain objects, referred to as *hubs*, are the nearest neighbors of many other objects. In the case of ZSL, a semantic prototype $\mathbf{s}$ being the nearest neighbors of many projections of visual samples $\hat{\mathbf{t}}_n$ into the semantic space is not desirable if $\mathbf{s}$ is not the prototype of the associated classes, as this will lead to predict the wrong label.

When using ridge regression for ZSL, it has been verified experimentally [78, 135] that this situation tends to happen. However, it is evidenced in [135] that this effect is mitigated when projecting from the semantic to the visual space, compared to projections from the visual to the semantic space.

It should be noted that the hubness problem does not occur exclusively when using ridge regression, and more complex ZSL methods such as [168] make use of the findings of [135].

Semantic autoencoder. The semantic autoencoder (**SAE**) [70] approach can be seen as a combination of the two approaches $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ and $\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$. The main idea consists in first *encoding* a visual sample $\mathbf{x}_n$ by linearly projecting it onto the semantic space with $\hat{\mathbf{t}}_n = \mathbf{W}\mathbf{x}_n$, and then *decoding* it by projecting the result into the visual space again with $\hat{\mathbf{x}}_n = \mathbf{V}\hat{\mathbf{t}}_n$, $\mathbf{W} \in \mathbb{R}^{K\times D}$ and $\mathbf{V} \in \mathbb{R}^{D\times K}$ being the parameters of the model.

While a standard autoencoder would only be concerned with the reconstruction error $\|\hat{\mathbf{x}}_n - \mathbf{x}_n\|^2$, we have an additional constraint since we want the encoded representation $\hat{\mathbf{t}}_n$ to correspond to the semantic representation $\mathbf{t}_n = \mathbf{s}_{y_n}$ of $\mathbf{x}_n$. This can be taken into account by adding another component $\|\hat{\mathbf{t}}_n - \mathbf{t}_n\|^2$ to the loss to measure the encoding error. To simplify the problem, we can use only one matrix of parameters $\mathbf{W}$ by setting $\mathbf{V} = \mathbf{W}^\top$. Using the same notations as previously, this results in the following loss:

$$\|\mathbf{X} - \mathbf{T}\mathbf{W}\|^2 + \gamma\|\mathbf{T} - \mathbf{X}\mathbf{W}^\top\|^2 \tag{1.41}$$

where γ is a hyperparameter and represents a trade-off between the reconstruction loss $\|\mathbf{X} - \mathbf{T}\mathbf{W}\|^2$ and the semantic encoding loss $\|\mathbf{T} - \mathbf{X}\mathbf{W}^\top\|^2$.

Using Lemma 1.26 to obtain the derivative of Equation (1.41) with respect to $\mathbf{W}$, setting it to 0

and rearranging, we get

$$\mathbf{T}^\top\mathbf{T}\mathbf{W} + \gamma\mathbf{W}\mathbf{X}^\top\mathbf{X} = (1-\gamma)\mathbf{T}^\top\mathbf{X} \tag{1.42}$$

Contrary to the previous examples, there is no immediate closed-form solution to this problem. However, if we define $\mathbf{A} = \mathbf{T}^\top\mathbf{T}$, $\mathbf{B} = \gamma\mathbf{X}^\top\mathbf{X}$ and $\mathbf{C} = (1-\gamma)\mathbf{T}^\top\mathbf{X}$, Equation (1.42) can be written $\mathbf{A}\mathbf{W} + \mathbf{W}\mathbf{B} = \mathbf{C}$. This corresponds to a Sylvester equation, and a numerical solution can be computed efficiently using the Bartels-Stewart algorithm [9].

During the testing phase, predictions can be made either in the semantic space using Equation (1.25) or in the visual space using Equation (1.40).

Non linear approaches. For now, all methods of this section used linear projections to predict features of one modality (visual or semantic) from the other. However, the losses in equations (1.24), (1.35), (1.38) or (1.41) could be easily adapted to non-linear regression methods.

One such example, **Cross-Modal Transfer** or **CMT**, is proposed in [138]. It consists in a simple fully-connected, 1-hidden layer neural network with hyperbolic tangent non-linearity, which is used to predict semantic prototypes from visual features. Equation (1.23) (and (1.24)) can therefore simply be re-written as

$$\frac{1}{N}\sum_n \|\mathbf{t_n} - \mathbf{W}_2\tanh(\mathbf{W}_1\mathbf{x}_n)\|_2^2 \tag{1.43}$$

$\mathbf{W}_1 \in \mathbb{R}^{H\times D}$ and $\mathbf{W}_2 \in \mathbb{R}^{K\times H}$ being the parameters of the model, and H being the dimension of the hidden layer which is a hyperparameter.

Similar or more complex adaptations can easily be made for other methods. The main drawback of such non linear projections compared to the linear methods presented earlier is that there is no general closed-form solution, and iterative numerical algorithms must be used to determine suitable values for the parameters.

1.2.3 Ranking methods

Ranking methods make a more direct use of the compatibility function f. The intuition behind these methods is that the compatibility of matching pairs should be much higher than the compatibility of non-matching pairs. More specifically, given a visual sample $\mathbf{x}$ with label y, we expect that its compatibility with the corresponding prototype $\mathbf{s}_y$ should be much higher than its compatibility with

$\mathbf{s}_c$, the prototype of a different class $c \neq y$:

$$f(\mathbf{x}, \mathbf{s}_y) \gg f(\mathbf{x}, \mathbf{s}_c),\ c \neq y \tag{1.44}$$

How "much higher" can be more precisely and quantitatively defined through the introduction of a margin m, such that $f(\mathbf{x}, \mathbf{s}_y) \geq m + f(\mathbf{x}, \mathbf{s}_c)$. This is equivalent to say that we want

$$m + f(\mathbf{x}, \mathbf{s}_c) - f(\mathbf{x}, \mathbf{s}_y) \leq 0 \tag{1.45}$$

To enforce this constraint, we can penalize triplets $(\mathbf{x}, \mathbf{s}_y, \mathbf{s}_c),\ c \neq y$ for which this inequality is not true using the *triplet loss*:

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{s}_c, \mathbf{s}_y; f) = [m + f(\mathbf{x}, \mathbf{s}_c) - f(\mathbf{x}, \mathbf{s}_y)]_+ \tag{1.46}$$

where $[\cdot]_+ = \max(0, \cdot)$. This way, for a given triplet $(\mathbf{x}, \mathbf{s}_y, \mathbf{s}_c),\ c \neq y$, the loss is 0 if $f(\mathbf{x}, \mathbf{s}_c)$ is much smaller than $f(\mathbf{x}, \mathbf{s}_y)$, and is all the higher as $f(\mathbf{x}, \mathbf{s}_c)$ gets close to – or surpasses – $f(\mathbf{x}, \mathbf{s}_y)$.

This triplet loss is also often called hinge rank loss, as it can be viewed as an extension of the standard hinge loss used with SVMs for binary classification. The hinge loss can be written $[1 - y \cdot f(\mathbf{x})]_+$, where y is a binary label in $\{-1, 1\}$ and f attributes a score to input $\mathbf{x}$, for example $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ with a linear kernel. This hinge loss can be modified to handle multi-class classification directly, without requiring strategies such as one-versus-one or one-versus-all. An example of such a modification is the Weston-Watkins [160] multi-class SVM loss:

$$\mathcal{L}_{\text{WW}}(\mathbf{x}, y; f) = \sum_{c \neq y} [1 + f_{\mathbf{w}_c}(\mathbf{x}) - f_{\mathbf{w}_y}(\mathbf{x})]_+ \tag{1.47}$$

This corresponds to the triplet loss in Equation (1.46) summed over all classes $c \neq y$, with $m = 1$ and a special case of f where $f_{\mathbf{w}}(\mathbf{x}) = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$.

In general, it is not possible to derive a solution analytically for methods based on a triplet loss, so we must resort to the use of numerical optimization. It should be noted that the use of the triplet loss is far from being restricted to ZSL, and is in fact widespread in other tasks requiring to learn compatibilities, such as image-sentence retrieval [63, 62, 156] or face recognition [133] among many others.

1.2.3.1 Linear compatibility function

In many triplet loss approaches to ZSL, the compatibility function f is simply defined as a bilinear mapping between the visual and semantic spaces parameterized by a matrix $\mathbf{W} \in \mathbb{R}^{D \times K}$:

$$f_{\mathbf{W}}(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \mathbf{W} \mathbf{s} \tag{1.48}$$

This compatibility function is actually the same as the one we encountered earlier with ESZSL, even though the loss function used to learn its parameters $\mathbf{W}$ is different.

The **Deep Visual-Semantic Embedding** model or **DeViSE** [40] is one of the simplest applications of a triplet loss with a linear compatibility function to ZSL: the total loss is simply the sum of the triplet loss over all training triplets $(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c),\ c \neq y$:

$$\mathcal{L}^{\text{tr}}(\mathcal{D}^{tr}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{\substack{c \in \mathcal{C}^{\mathcal{S}} \\ c \neq y_n}} [m + f(\mathbf{x}_n, \mathbf{s}_c) - f(\mathbf{x}_n, \mathbf{s}_{y_n})]_+ \tag{1.49}$$

DeViSE can also be viewed as a direct application of the Weston-Watkins loss [160] to ZSL. It can be noted that the link with the generic loss framework in Equation (1.3) is this time pretty straightforward, as with many triplet loss methods. Although no explicit regularization Ω on f is mentioned in the original publication – even though the authors make use of early stopping in the gradient descent – it is again straightforward to add an $\ell 2$ penalty $\Omega[f_{\mathbf{W}}] = \|\mathbf{W}\|_2^2$.

The **Structured Joint Embedding** approach, or **SJE** [4], is fairly similar to DeViSE. It is inspired by works on structured SVMs [148, 149], and makes use of the Cramer-Singer loss [30] for multi-class SVM. The general formulation for the Crammer-Singer loss is

$$\mathcal{L}_{\text{CS}}(\mathbf{x}, y; f) = [\max_{c \neq y} \ (m + f_{\mathbf{w}_c}(\mathbf{x}) - f_{\mathbf{w}_y}(\mathbf{x}))]_+ \tag{1.50}$$

Applied to a ZSL setting, this means that only the class which is violating the constraint in Equation (1.45) the most is taken into account for each sample. In our case, this results in

$$\mathcal{L}^{\text{tr}}(\mathcal{D}^{\text{tr}}) = \frac{1}{N} \sum_{n=1}^{N} \max_{\substack{c \in \mathcal{C}^{\mathcal{S}} \\ c \neq y_n}} ([m + f(\mathbf{x}_n, \mathbf{s}_c) - f(\mathbf{x}_n, \mathbf{s}_{y_n})]_+) \tag{1.51}$$

The **Attribute Label Embedding** approach or **ALE** [2, 3] considers the ZSL task as a ranking problem, where the objective is to rank the correct class c as high as possible on the list of possible candidate unseen classes. From this perspective, we can consider that SJE only takes into account the top element of the ranking list provided the margin m is close to 0. By contrast, DeViSE penalizes all ranking mistakes: given labeled sample $(\mathbf{x}, y)$, for all classes c mistakenly ranked higher than y, we have $f(\mathbf{x}, \mathbf{s}_c) > f(\mathbf{x}, \mathbf{s}_y)$ which contributes to the loss. The ALE approach aims to be somewhere in between these two approaches, so that a mistake on the rank when the true class is close to the top of the ranking list weighs more than a mistake when the true class is lower on the list.

Inspired by works on weighted approximate ranking [159, 150], the authors of [2] define

$$r(\mathbf{x}, y) = 1 + \sum_{c,\ c \neq y} \mathbb{1}[\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{s}_c, \mathbf{s}_y) > 0] \tag{1.52}$$

the number of triplets violating the constraint in Equation(1.45), which is an upper bound on the rank of label y for image $\mathbf{x}$. They further define

$$\gamma(r) = \frac{1}{r} \sum_{i=1}^{r} \frac{1}{i} \tag{1.53}$$

which is a slowly decreasing function of r. The final total loss is

$$\mathcal{L}^{\text{tr}}(\mathcal{D}^{tr}) = \frac{1}{N} \sum_{n} \gamma(r(\mathbf{x}_n, y_n)) \sum_{\substack{c \in \mathcal{C}^{\mathcal{S}} \\ c \neq y_n}} [m + f(\mathbf{x}, \mathbf{s}_c) - f(\mathbf{x}, \mathbf{s}_y)]_+ \tag{1.54}$$

which is similar to DeViSE but decreases the contribution to the loss for training samples with many ranking mistakes. This prompts the model to prioritize images whose correct label is already close to the top of the ranking list.

As a side note, each of [40, 4, 2] proposes more contributions in the original article than is presented here. For example, DeViSE uses fine-tuning on the convolutional network, SJE enables the use of several semantic prototypes, ALE takes attributes correlation into account and proposes an extension to few-shot learning. However, we are mostly interested in providing a general overview of the use of triplet loss in ZSL, and we refer the reader to the original articles for further details.

1.2.3.2 Non linear compatibility function

Similarly to CMT in the previous section, all models above can be extended to be non linear; such an extension is even more straightforward as this time, having no closed-form solution, all models require

the use of numerical optimization. One such example of a non linear model worth describing due to its historical significance and still reasonable performance is the **Synthesized Classifiers** approach, or **SynC** [22, 23]. Based on a manifold learning framework, it aims to learn *phantom classes* in both the semantic and visual spaces, so that linear classifiers for seen and unseen classes can be synthesized as a combination of such phantom classes.

More precisely, the goal is to synthesize linear classifiers $\mathbf{w}_c$ in the visual space such that the compatibility between image $\mathbf{x}$ and class c can be computed with $f(\mathbf{x}, \mathbf{s}_c) = \mathbf{w}_c^\top \mathbf{x}$. This will lead to predict

$$\hat{y} = \underset{c}{\operatorname{argmax}}\ \mathbf{w}_c^\top \mathbf{x} \tag{1.55}$$

Let's call respectively $\{\overset{*}{\mathbf{x}}_p\}_{p\in[\![1,P]\!]}$ and $\{\overset{*}{\mathbf{s}}_p\}_{p\in[\![1,P]\!]}$ the P phantom classes in the respective visual and semantic spaces. These phantom classes are learned and constitute the parameters of the model[5]. Each visual classifier will be synthesized as a linear combination of visual phantom classes:

$$\mathbf{w}_c = \sum_{p=1}^{P} v_{c,p}\ \overset{*}{\mathbf{x}}_p \tag{1.56}$$

The value of each coefficient $v_{c,p}$ is set so as to correspond to the conditional probability of observing phantom class $\overset{*}{\mathbf{s}}_p$ in the neighborhood of real class $\mathbf{s}_c$ in the semantic space. Following works on manifold learning [57, 96], this can be estimated with

$$v_{c,p} = \frac{\exp\left(-\|\mathbf{s}_c - \overset{*}{\mathbf{s}}_p\|^2/2\sigma^2\right)}{\sum_q \exp\left(-\|\mathbf{s}_c - \overset{*}{\mathbf{s}}_q\|^2/2\sigma^2\right)} \tag{1.57}$$

The parameters of the model, i.e. the phantom classes $\{(\overset{*}{\mathbf{x}}_p, \overset{*}{\mathbf{s}}_p)\}_p$, can be estimated by making use of the Crammer-Singer loss[5] similarly to equations (1.50) and (1.51), and using adequate regularization to obtain the following objective:

$$\underset{\{(\overset{*}{\mathbf{x}}_p, \overset{*}{\mathbf{s}}_p)\}_p}{\text{minimize}}\ \frac{1}{N}\sum_{n=1}^{N} \max_{\substack{c\in\mathcal{C}^\mathcal{S}\\ c\neq y_n}} \Big([m + \mathbf{w}_c^\top \mathbf{x}_n - \mathbf{w}_{y_n}^\top \mathbf{x}_n]_+\Big) + \Omega \tag{1.58}$$

$$\Omega = \lambda \sum_{c\in\mathcal{C}^\mathcal{S}} \|\mathbf{w}_c\|^2 + \gamma \sum_{p=1}^{P} \|\overset{*}{\mathbf{s}}_p\|^2 \tag{1.59}$$

[5] A number of simplifications were made for the sake of clarity and brevity: in the original article [22], phantom classes are actually sparse linear combinations of semantic prototypes, $v_{c,p}$ can further use Mahalanobis distance, other losses such as squared hinge loss can be used instead of the Crammer-Singer loss, euclidean distances between semantic prototypes can be used instead of a fixed margin in the triplet loss, additional regularization terms and hyperparameters are introduced, and optimization between $\{\overset{*}{\mathbf{x}}_p\}_p$ and $\{\overset{*}{\mathbf{s}}_p\}_p$ is performed alternatingly.

where λ and γ are hyperparameters along with σ^2 in Equation (1.57).

It is interesting to note that ALE can actually be considered as a special case of SynC, where the classifiers are simply a linear combination of semantic prototypes $\mathbf{w}_c = \mathbf{W}\mathbf{s}_c$.

Other methods making use of a triplet loss include [28], which uses multiple training objectives and is based on an architecture inspired by adversarial autoencoders [97]. [90] applies an extension of the Crammer-Singer loss which enables the use of unlabeled samples during training.

1.2.4 Generative methods

Generative methods aim to generate visual samples belonging to unseen classes based on their semantic description; these samples can then be used to train standard classifiers. Partly for this reason, most generative methods directly generate high-level visual features, as opposed to raw pixels – another reason being that generating raw images is usually not as effective [162].

Generative methods have gained a lot of attention in the last few years: many if not most recent high-visibility ZSL approaches are generative models [152, 162, 164]. This is partly because such approaches have interesting properties, which make them particularly suitable to certain settings such as Generalized ZSL (Section 1.4). However, a disadvantage of these approaches is that they can usually only operate in a class-transductive setting, since the class prototypes of unseen classes are needed to generate samples belonging to these classes; contrary to methods based on regression or explicit compatibility functions, additional training is often necessary to integrate new classes to the model.

We divide generative methods into two main categories: methods generating a parametric distribution, which consider visual samples behave according to a standard probability distribution such as a multivariate Gaussian and try to estimate its parameters so that visual features can be sampled from this distribution, and non-parametric methods, where visual samples are directly generated by the model.

Whether $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ is a generative method can be debated, as it predicts visual features from semantic descriptions. However, since in the standard ZSL setting, a single prototype is available for each class, $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ can only generate a single visual sample per class. On the other hand, we expect a generative method to be able to produce many visual instances. For this reason, we consider that $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ or other similar methods are not generative models.

1.2.4.1 Parametric distribution

Methods in this category assume that visual features for each class follow a standard parametric distribution. For example, we may assume that for each class c, visual features are samples from a multivariate Gaussian with mean $\boldsymbol{\mu}_c \in \mathbb{R}^D$ and covariance $\boldsymbol{\Sigma}_c \in \mathbb{R}^{D \times D}$ (with the constraint that $\boldsymbol{\Sigma}_c$ must be positive semi-definite). This results in the following probability density function for samples $\mathbf{x}$ from class c:

$$p(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{1.60}$$

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{1.61}$$

If we can estimate these parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for unseen classes, we will be able to generate samples belonging to these classes. Zero-shot recognition can then be performed by training a standard multi-class classifier – usually a softmax as in [17, 162] or an SVM as in [49, 152] – on the labeled generated samples.

Alternatively, once the probability density functions' parameters have been estimated for unseen classes, we may determine the class of a test visual sample $\mathbf{x}$ using maximum likelihood or similar methods [153]:

$$\hat{y} = \underset{c \in \mathcal{C}^\mathcal{U}}{\operatorname{argmax}}\ p(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{1.62}$$

Other approaches [162] also propose to further train a ZSL model based on an explicit compatibility function using the generated samples and the corresponding class prototypes, and then perform zero-shot recognition as usually with Equation (1.4).

The **Generative Framework for Zero-Shot Learning** or **GFZSL** [153] assumes that visual features are normally distributed given their class as in Equation (1.60). To simplify, we can further assume that $\boldsymbol{\Sigma}_c = \operatorname{diag}(\boldsymbol{\sigma}_c^2)$ with $\boldsymbol{\sigma}_c^2 \in \mathbb{R}^{+D}$. The parameters of the distribution $(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2)$ are easy to estimate for seen classes $c \in \mathcal{C}^\mathcal{S}$. For example, using maximum likelihood estimators:

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{M_c} \sum_{m=1}^{M_c} \mathbf{x}_m^c \tag{1.63}$$

$$\hat{\boldsymbol{\sigma}}_c^2 = \frac{1}{M_c} \sum_{m=1}^{M_c} (\mathbf{x}_m^c - \hat{\boldsymbol{\mu}}_c) \odot (\mathbf{x}_n^c - \hat{\boldsymbol{\mu}}_c) \tag{1.64}$$

where $\{\mathbf{x}_1^c, \dots, \mathbf{x}_{N_c}^c\}$ are the M_c visual samples with label $y = c$, and $\odot$ is the Hadamard (element-wise) product.

However, these parameters are unknown for unseen classes. Since the only information we have about unseen classes is the class prototypes, we can – or have to – assume that the parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$ of class c depend on class prototype $\mathbf{s}_c$. [153] further assumes a linear dependency, such that

$$\boldsymbol{\mu}_c = \mathbf{W}_\mu^\top \mathbf{s}_c \tag{1.65}$$

$$\boldsymbol{\rho}_c = \log(\boldsymbol{\sigma}_c^2) = \mathbf{W}_\sigma^\top \mathbf{s}_c \tag{1.66}$$

The models' parameters $\mathbf{W}_\mu \in \mathbb{R}^{K \times D}$ and $\mathbf{W}_\sigma \in \mathbb{R}^{K \times D}$ can then be obtained using ridge regression, using the class distribution parameters $\{(\hat{\boldsymbol{\mu}}_c, \hat{\boldsymbol{\rho}}_c)\}_{c \in \mathcal{C}^\mathcal{S}}$ estimated on seen classes as training samples. In a similar way to what we did in Section 1.2.2, we write $\mathbf{M} = (\hat{\boldsymbol{\mu}}_1, \dots, \hat{\boldsymbol{\mu}}_C)^\top \in \mathbb{R}^{C \times D}$ and $\mathbf{R} = (\hat{\boldsymbol{\rho}}_1, \dots, \hat{\boldsymbol{\rho}}_C)^\top \in \mathbb{R}^{C \times D}$ the regression targets, with $C = |\mathcal{C}^\mathcal{S}|$ the number of seen classes. We want to minimize regularized $\ell 2$ losses

$$\|\mathbf{M} - \mathbf{S}\mathbf{W}_\mu\|^2 + \lambda_\mu \|\mathbf{W}_\mu\|^2 \tag{1.67}$$

$$\|\mathbf{R} - \mathbf{S}\mathbf{W}_\sigma\|^2 + \lambda_\sigma \|\mathbf{W}_\sigma\|^2 \tag{1.68}$$

λ_μ and λ_σ being hyperparameters.

Equations (1.67) and (1.68) are of the same form as Equation(1.38) in Section 1.2.2, so solutions are of the same form as Equation (1.39):

$$\mathbf{W}_\mu = (\mathbf{S}^\top \mathbf{S} + \lambda_\mu \mathbf{I}_K)^{-1} \mathbf{S}^\top \mathbf{M} \tag{1.69}$$

$$\mathbf{W}_\sigma = (\mathbf{S}^\top \mathbf{S} + \lambda_\sigma \mathbf{I}_K)^{-1} \mathbf{S}^\top \mathbf{R} \tag{1.70}$$

We can then predict parameters $(\hat{\boldsymbol{\mu}}_c, \hat{\boldsymbol{\rho}}_c)$ for all unseen classes $c \in \mathcal{C}^\mathcal{U}$, and sample visual features of unseen classes accordingly. Predictions can then be made using either a standard classifier or the estimated distributions themselves using Equation (1.62) as detailed previously.

[153] also propose to extend this approach to include more generic distributions belonging to the exponential family and non-linear regressors.

The **Synthesized Samples for Zero-Shot Learning** or **SSZSL** [49] approach similarly assumes that $p(\mathbf{x}|c)$ is gaussian, estimates parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for seen classes with techniques similar to GFZSL (equations (1.63) and (1.64)) and aims to predict parameters $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ for unseen classes. We can again assume that $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$. In a way that reminds the ConSE method, the distributions parameters are estimated using a convex combination of parameters from seen classes d:

$$\hat{\boldsymbol{\mu}} = \frac{1}{Z} \sum_{d \in \mathcal{C}^{\mathcal{S}}} w_d \boldsymbol{\mu}_d \tag{1.71}$$

$$\hat{\boldsymbol{\sigma}}^2 = \frac{1}{Z} \sum_{d \in \mathcal{C}^{\mathcal{S}}} w_d \boldsymbol{\sigma}_d^2 \tag{1.72}$$

with $Z = \mathbf{1}^\top \mathbf{w} = \sum_d w_d$. The model therefore has one parameter $\mathbf{w}_c \in \mathbb{R}^{|\mathcal{C}^{\mathcal{S}}|}$ to determine per unseen class c.

These model parameters $\mathbf{w}_c$ are set such that the semantic prototype $\mathbf{s}_c^{\text{te}}$ from unseen class c is approximately a convex combination of prototypes from seen classes, i.e. $\mathbf{s}_c^{\text{te}} \simeq \mathbf{S}^\top \mathbf{w}_c / Z_c$, while preventing classes dissimilar to $\mathbf{s}_c^{\text{te}}$ from being assigned a large weight. This results in the following loss for unseen class c:

$$\|\mathbf{s}_c^{\text{te}} - \mathbf{S}^\top \mathbf{w}_c\|_2^2 + \lambda \mathbf{w}_c^\top \mathbf{d}_c \tag{1.73}$$

where each element $(\mathbf{d}_c)_i$ of $\mathbf{d}_c$ is a measure of how dissimilar[6] unseen class c is to seen class i, and λ is a hyperparameter. Minimizing the second term in Equation (1.73) will naturally lead to assign smaller weights to classes dissimilar to c. Using Lemma 1.26, we get the closed-form solution

$$\mathbf{w}_c = (\mathbf{S}\mathbf{S}^\top)^{-1} (\frac{\lambda}{2} \mathbf{d}_c - \mathbf{S}\mathbf{s}_c^{\text{te}}) \tag{1.74}$$

Other examples of methods which assume that class-conditional distribution of visual samples is gaussian and estimate the corresponding parameters using class prototypes include [155], which models locally linear dependencies between visual and semantic spaces and makes use of sparse coding.

1.2.4.2 Non parametric distribution

As explained previously, such approaches do not explicitly make simplifying assumptions about the shape of the distribution of visual features, and use powerful generative methods such as varia-

[6] In [49], the authors use $(\mathbf{d}_c)_i = \left(\exp\left(-\frac{\|\mathbf{s}_c^{\text{te}} - \mathbf{s}_i^{\text{tr}}\|^2}{\bar{\alpha}^2} \right) \right)^{-1}$ to measure how dissimilar unseen class c is to seen class i, where $\bar{\alpha}$ is the mean value of the distances between any two prototypes from seen classes.

tional auto-encoders (VAEs) [66] or generative adversarial networks (GANs) [47] to directly synthesize samples. Although these models are in principle able to capture complex data distribution, they can prove to be hard to train [6].

The **Synthesized Examples for GZSL**, or **SE-GZSL** [152] is an example of an approach based on a conditional VAE [139] architecture. It consists of two main parts: an *encoder* $\mathcal{E}(\cdot)$ which maps an input $\mathbf{x}$ to an R-dimensional internal representation or latent code $\mathbf{z} \in \mathbb{R}^R$, and a *decoder* $\mathcal{D}(\cdot)$ which tries to reconstruct the input $\mathbf{x}$ from the internal representation. An optional third part can be added to the model: a *regressor* $\mathcal{R}(\cdot)$ which estimates the semantic representation $\mathbf{t}$ of the visual input $\mathbf{x}$. All components consist of multi-layer feedforward networks.

To be more precise, the encoder learns a *distribution* over the internal representations, and the latent code $\mathbf{z}$ is randomly sampled from this distribution. We usually assume it is a gaussian $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, such that the encoder predicts the estimated parameters[7] $\hat{\boldsymbol{\mu}} \in \mathbb{R}^R$ and $\hat{\boldsymbol{\sigma}}^2 \in \mathbb{R}^{+R}$ based on the input $\mathbf{x}$: $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^R \times \mathbb{R}^{+R}$. We will still write $\mathbf{z} = \mathcal{E}(\mathbf{x})$ to indicate a representation $\mathbf{z}$ sampled from the distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2)$ regressed from $\mathbf{x}$ by $\mathcal{E}$. Similarly, the decoder and regressor output parameters of probability distributions, which we also assume to be normal.

To help the decoder to produce class-dependant reconstructed outputs, the corresponding class prototype $\mathbf{t}_n = \mathbf{s}_{y_n}$ is concatenated to the representation $\mathbf{z}_n$ for input $\mathbf{x}_n$. The decoder is then a function $\mathcal{D} : \mathbb{R}^R \times \mathbb{R}^K \to \mathbb{R}^D \times \mathbb{R}^{+D}$. We will write $\hat{\mathbf{x}}_n = \mathcal{D}(\mathcal{E}(\mathbf{x_n}), \mathbf{t}_n)$ the (sampled) reconstructed input.

The loss of the model is the (conditional) variational lower-bound [66, 139]:

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = -\mathbb{E}_{p(\mathbf{z}|\mathbf{x})}[\log(p(\mathbf{x}|\mathbf{z}, \mathbf{t}))] + \mathbb{KL}[p(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \tag{1.75}$$

where $\mathbb{E}_{p(\mathbf{x})}[g(\mathbf{x})]$ denotes the expectation of quantity $g(\mathbf{x})$ with respect to distribution $p(\mathbf{x})$, and $\mathbb{KL}[p||q]$ is the Kullback-Leibler divergence [73] of distributions p and q. In our case, since the encoder outputs a probability distribution with parameters $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2)$ from $\mathbf{x}$, we have $p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2)$. The same idea can be applied to the decoder and $p(\hat{\mathbf{x}}|\mathbf{z}, \mathbf{t})$. The prior $p(\mathbf{z})$ over the latent code is usually assumed to be a unit gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Given the forms of $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z})$, their KL divergence has

[7] We actually predict $\hat{\boldsymbol{\rho}} \in \mathbb{R}^R$ such that $\hat{\boldsymbol{\sigma}}^2 = \exp(\hat{\boldsymbol{\rho}}) \in \mathbb{R}^{+R}$ as in Equation (1.66).

a closed-form solution:

$$\mathbb{KL}[p(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] = -\frac{1}{2}\sum_{r=1}^{R}(\mathbf{1} + \log(\boldsymbol{\sigma}^2) - \boldsymbol{\mu}^2 - \boldsymbol{\sigma}^2)_r \tag{1.76}$$

Other approaches such as [105] consider that only the encoder outputs a probability distribution. In the latter case, we may assume that the true distribution of visual samples is an isotropic gaussian given the latent representation, i.e. $p(\mathbf{x}|\mathbf{z},\mathbf{t}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}(\mathbf{z},\mathbf{t}), \sigma^2\mathbf{I})$. In this case, the output of the decoder should be $\hat{\mathbf{x}} = \boldsymbol{\mu}(\mathbf{z},\mathbf{t})$, and it can be shown that minimizing $-\log(p(\mathbf{x}|\mathbf{z},\mathbf{t}))$ is equivalent to minimizing $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$. Furthermore, in [105], the class prototype is appended to the visual sample as opposed to the latent code. This results in the following training loss:

$$\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \mathcal{D}(\mathcal{E}(\mathbf{x}_n,\mathbf{t}_n))\|_2^2 + \lambda\|\boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - \mathbf{1}\|_1 \tag{1.77}$$

where λ is a hyper-parameter.

The authors of [152] further propose to use the regressor $\mathcal{R}$ to encourage the decoder to generate discriminative visual samples, by adding other weighted components to the loss in Equation (1.75). An example of such components consists in evaluating the quality of predicted attributes from synthesized samples, and takes the form $\mathcal{L} = -\mathbb{E}_{p(\hat{\mathbf{x}}|\mathbf{z},\mathbf{t})p(\mathbf{z})p(\mathbf{t})}[\log(p(\mathbf{a}|\hat{\mathbf{x}}))]$. The regressor itself is trained on both labeled training samples and generated samples, and the parameters of the encoder / decoder and the regressor are optimized alternatingly.

f-GAN [162] is based on a similar approach, but makes use of conditional GANs [104] to generate visual features. It consists of two parts: a *discriminator* $\mathcal{D}$ which tries to distinguish real images from synthesized images, and a *generator* $\mathcal{G}$ which tries to generate images that $\mathcal{D}$ cannot distinguish from real images. Both parts consist in a multi-layer neural network.

The generator is similar to the decoder from the previous approach in that it takes as input a latent code $\mathbf{z} \in \mathbb{R}^R$ and the semantic representation $\mathbf{s}_c$ of a class c, and tries to generate a visual sample $\hat{\mathbf{x}}$ of class c: $\mathcal{G} : \mathbb{R}^R \times \mathbb{R}^K \rightarrow \mathbb{R}^D$. The key difference is that the latent code is not the output of an encoder but consists of random gaussian noise. The discriminator takes as input a visual sample, either real or generated, of a class c as well as the prototype $\mathbf{s}_c$, and predicts the probability that the visual sample was generated: $\mathcal{D} : \mathbb{R}^D \times \mathbb{R}^K \rightarrow [0,1]$.

$\mathcal{G}$ and $\mathcal{D}$ both compete in a two-player minimax game, such that the optimization objective is:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{p(\mathbf{x},y),p(\mathbf{z})}[\log(\mathcal{D}(\mathbf{x},\mathbf{s}_y)) + \log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z},\mathbf{s}_y),\mathbf{s}_y))] \tag{1.78}$$

The authors of [162] further propose to train an improved Wasserstein GAN [7, 48], and similarly to [152], they add another component to the loss to ensure that generated features are discriminative, using a classification loss instead of a regression loss. They call this extended approach f-CLSWGAN.

Several other generative approaches have been proposed. [17, 18] notably explore many such approaches, such as generative moment matching networks [91], standard conditional generative adversarial networks [47, 110], denoising auto-encoders [11] and adversarial auto-encoders [97]. f-VAEGAN-D2 [164] proposes a combination of VAE and GAN, which is also applicable to few-shot learning.

1.3 Visual and semantic representations

As mentioned in Section 1.1.2, usual ZSL methods employ high-level visual features denoted $\mathbf{x}$ to represent images, as well as a single semantic prototype per class denoted $\mathbf{s}$. Both have a fixed number of dimensions, which we call D for the dimension of the visual space and K for the dimension of the semantic space. In this section, we provide an overview of the main approaches used to obtain such representations. We also briefly present a few alternative settings were additional of different information may be used.

1.3.1 Visual features

As mentioned earlier, in the ZSL task, images are usually represented by vectors of high-level visual features, as opposed to raw pixels. In the early days of ZSL [75], these high-level visual representations were obtained with "hand-crafted" visual feature extractors such as Scale Invariant Feature Transform (SIFT) [94], Speeded Up Robust Features (SURF) [10] or Pyramid of Histograms of Orientation Gradients (PHOG) [15]. Since then, the computer vision field has embraced the deep learning revolution [83], and deep convolutional neural networks are now used for many if not most computer vision tasks [72, 136]. These networks can be either trained from scratch for a specific task, or trained for a source task within a source domain with the aim of being used later on a target task and a target domain, according to a transfer learning paradigm [145].

In the field of zero-shot learning, it is thus conventional to use a deep convolutional neural network trained on a generic image classification task as a fixed feature extractor [40, 108, 161]. Examples of deep architectures often used as feature extractors for zero-shot learning – or as a matter of fact for other tasks [156, 145] – include AlexNet [72], VGG [136], GoogLeNet [144] or ResNet [55]. These networks are typically trained on image classification tasks with datasets containing a large number of diverse classes, such as the ImageNet dataset [33] with the 1000 classes from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [132]. As a side note, most major deep learning frameworks [113, 1] include pre-trained versions of common architectures, so that it is usually not necessary to actually train these networks. For a given image, high-level visual features can then be extracted from the internal representations resulting from a forward pass of the data in the network. These features typically correspond to the penultimate activations of the network, i.e. the internal representation just before the last fully connected layer producing activation outputs for each class. For AlexNet and VGG, these representations correspond to the 4096-dimensional activations after the second fully connected layers, themselves located after the convolutional layers; for GoogLeNet, these are the 1024-dimensional activations of the pooling units. The most common visual features extractor in recent ZSL works [5, 162, 53, 23] is now the ResNet-101 [55] architecture, in which the extracted features are the 2048-dimensional activations of the last pooling layer.

It is of course also possible to train a deep learning architecture from scratch on images from the ZSL dataset, provided we only use images from the seen classes [142] – and also do not use test images from seen classes in a generalized ZSL setting, Section 1.4. Some works [92, 115] adopt a hybrid approach and fine-tune pre-trained networks either on a classification task on images from seen classes or in an end-to-end fashion, but this approach is not common in recent literature as it may hinder the reproducibility of ZSL results. Other works [126, 70, 28][8] further perform variants of "10-crop", which consists in extracting visual features for an image cropped on its middle, lower left, upper left, lower right and upper right parts as well as the equivalent for the horizontally-flipped image, and averaging the resulting visual features to obtain the final visual features corresponding to the image.

Finally, it is important to mention that as evidenced in [163, 161], using neural networks pre-trained on generic image datasets as visual feature extractors may induce a serious bias on some

[8]The use of 10-crop is not explicitly mentioned in [70], but the authors of [161] explain that they confirmed with the authors of [70] the use of 5 random crops from images.

datasets. Indeed, 6 of the 10 unseen test classes of the Animals with Attributes (AwA) dataset [75] (Appendix A.1) are also classes in the ImageNet dataset used to train most standard visual feature extractors, and similar issues are present in other datasets such as the Attributes Pascal Yahoo (aPY) dataset [37]. As a result, these classes cannot be considered as truly unseen since some instances have been "seen" by the visual feature extractor. It is therefore important to either select test classes for ZSL benchmark datasets so that they are not present in ImageNet, or use different datasets which do not induce this bias for pre-training visual feature extractors. The first solution has been largely embraced by the community, and recent ZSL works [5, 18, 162, 23] overwhelmingly employ the *"proposed splits"* from [161] as train/test splits. Previously reported results which use *"standard splits"* should additionally be considered to be biased.

1.3.2 Semantic representations

In the example from Section 1.1.2, we assumed the semantic prototype $\mathbf{s}_c$ associated with class c consisted of attributes such as *"is orange"*, *"has stripes"* or *"has hooves"* and *"has a long neck"*. Such prototypes can be either binary, in which case the class *tiger* may be represented by $(1\ 1\ 0\ 0)^\top$, or continuous, for example in $[0, 1]$, in which case the representation of *tiger* may be $(0.8\ 0.9\ 0.0\ 0.2)^\top$ to indicate among other things that a typical tiger is not fully orange, or that not *all* tigers are orange as there exist white tigers.

Nonetheless, these 4-attribute examples are quite simplistic, as standard ZSL datasets such as CUB [154] may have up to hundreds of attributes per class (Appendix A.1), and performance of usual ZSL models may drop quickly when fewer attributes are used, as will be studied in Section 3.2.4. However, in a large scale ZSL setting [129] with datasets such as ImageNet [33] with up to thousands of classes (Appendix A.1), it may not be practical to manually provide hundreds of attributes for thousands of classes. Although attribute selection methods have been proposed, they typically require to keep at least half of the initial attributes for usual ZSL datasets [50]. In addition, for an open generic dataset where novel classes are expected to appear over time, it may even be impossible to define a relevant set of attributes *a priori*. It may therefore be preferable to use different sources of semantic information in large scale settings.

Unsupervised prototypes. It has been proposed in [129, 40] to use *word embeddings* [131, 102] as a source of semantic information. Similarly to the visual feature extractors which can be trained on large generic image datasets (Section 1.3.1), these word embeddings can be trained on large generic text corpora such as Wikipedia. In addition, the models producing these word embeddings can be trained in an unsupervised way, and thus require almost no human annotation effort. For example, given a sequence of T words $\{w_1, \dots, w_T\}$, the skip-gram [100] architecture aims to find word representations which enable to predict the context, i.e. the surrounding words, of a word w_t in this sequence. This can be achieved by minimizing

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-S\leq i\leq S\\ i\neq 0}}\log p(w_{t+i}|w_t) \tag{1.79}$$

where S is the size of the context window. To obtain the word embeddings from this objective, each unique word w is associated with an "input" vector $\mathbf{v}_w$ and an "output" vector $\mathbf{v}'_w$, and $p(w_i|w_t)$ is computed such that

$$p(w_i|w_t) = \frac{\exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_{w_i})}{\sum_w \exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_w)} \tag{1.80}$$

The input vector representation $\mathbf{v}_w$ can then be used as the embedding of word w.

Different approaches have additionally been proposed to address shortcomings of the (skip-gram) Word2Vec [100, 102] method we just described. For instance, [102] proposes to use negative sampling as well as subsampling of frequent words to speed up training and obtain better word representations. GloVe [117] uses matrix factorization on a global word co-occurrence matrix to leverage global statistics of a corpus in addition to local contexts. FastText [14] can leverage "subword" information by computing vector representations for character n-grams instead of whole words.

As these embedding models provide a fixed-size K-dimensional representation for a given word, it is then possible to use the embedding corresponding to a class name, for instance the word embedding of *"tiger"*, as the class prototype. For classes whose name consists of several words, for instance *"white tiger"*, the embeddings of the two words "white" and "tiger" can be averaged to obtain the class prototype. Likewise, for classes consisting of several multi-word lemmas as is the case for the ImageNet dataset, the word embeddings from each lemma can be averaged to obtain lemma representations, which are then averaged to obtain class prototypes [54, 53]. Similarly to attribute-based prototypes (and sometimes visual features), class prototypes may further be "$\ell 2$-normalized" to obtain representations with unit $\ell 2$ norm [167].

Graph-based information. Some ZSL models further make use of auxiliary information in addition to the fixed-dimensional class prototypes $\mathbf{s}$ when such information is available. For instance, [129, 158, 60] make use of the existing hierarchical relations between classes available for certain datasets such as ImageNet (more details are provided in Appendix A.1). More specifically, [158] and [60] are based on Graph Convolutional Networks (GCN) [31, 67], and use an adjacency matrix $\mathbf{A} \in \mathbb{R}^{C \times C}$ defining graph relations between the C seen and unseen classes in addition to the "base" semantic representations $\mathbf{S} \in \mathbb{R}^{C \times K}$. This additional information is used by an L-layer fully connected neural network with activation weights $\mathbf{H}_l$ at layer $l \in [\![1, L]\!]$ defined such that

$$\mathbf{H}_l = \sigma\left(\mathbf{A}\mathbf{H}_{l-1}\mathbf{W}_l\right) \tag{1.81}$$

where $\mathbf{W}_l$ are learned parameters and σ is a non linear activation function, and $\mathbf{H}_0 = \mathbf{S}$. [158] and [60] use derivatives of this architecture, with the objective that the rows of the last activation matrix $\mathbf{H}_L \in \mathbb{R}^{C \times D}$ corresponding to seen classes match the weights of pre-trained linear visual classifiers. The rows of $\mathbf{H}_L$ corresponding to unseen classes can then be used as classifiers for these unseen classes. On the other hand, [95] uses *only* relations between classes by first computing a pairwise distance matrix $\mathbf{D}$ between classes using path lengths in the WordNet hierarchy as distances, and then performing classical multidimensional scaling on $\mathbf{D}$ to obtain semantic prototypes as the rows of the resulting matrix.

Importantly, it has been recently evidenced in [53] that many classes previously used as unseen test classes in the ImageNet dataset are subcategories or supercategories of seen classes which, among other problems affecting this benchmark, induces an important bias for ZSL evaluation. In particular, this tends to confer an unfair advantage to methods making use of hierarchical relations between classes. Similarly to the new "proposed splits" from [163] for AwA [75], [53] proposed to use new test classes for the ImageNet ZSL benchmark, which do not suffer from these problems. As a consequence, large scale results anterior to [53] should be considered to be biased.

Other methods may also use different types of additional information. For instance, [166] considers the task of identifying an object in an image, and uses *context* in addition to the visual appearance of the object and its semantic description, where context is defined as the set of objects appearing in the same image. Bounding boxes are provided for these objects. This context enables the use of contextual priors, based on the idea that next to a knife and a plate, an apple is more likely to appear

than a tennis ball. [98] is based on the same idea, but instead employs co-occurrence statistics of objects appearing together in multi-labeled images.

Instead of using fixed-dimensional semantic representations, other methods employ information which requires additional processing. For instance, [35, 89, 122, 36, 170] use detailed descriptions in natural language, extracted from the Wikipedia pages corresponding to the classes from the CUB and Flower datasets (Appendix A.1). Raw vector representations can be extracted using a simple bag-of-words approach [122] or Term Frequency-Inverse Document Frequency (TF-IDF) [35, 89, 36, 170], before being further processed by a multi-layer perceptron [89], dimensionality reduction [35, 36] or noise reduction through the $\ell_{2,1}$ regularization [109] of a linear transformation [122]. Instead of one prototype per class, [126] takes advantage of 10 short sentences *per image*, collected for the CUB and Flower datasets, to train a character-level neural language model from scratch in an end-to-end fashion.

1.4 Generalized zero-shot learning

As described in Section 1.1.4, in a Generalized Zero-Shot Learning (GZSL) setting, test instances can belong to either a seen or an unseen class. As GZSL is a central aspect in later sections, especially in Chapter 2, we provide more details on this specific setting in this section. In particular, we present different approaches to the problem of GZSL, emphasize some problems specific to this setting, and introduce standard metrics to measure ZSL and GZSL scores.

Approaches to extend zero-shot recognition to generalized zero-shot recognition can be divided into roughly two categories: (1) approaches which explicitly try to identify when a sample does not belong to a seen class, and use either a standard classifier or a ZSL method depending on the result, and (2) approaches which use a unified framework for both seen and unseen classes.

Dealing with seen and unseen classes. In [138], the authors explicitly estimate $g_u(\mathbf{x}) = P(y \in \mathcal{C}^{\mathcal{U}}|\mathbf{x})$, the probability that a test instance $\mathbf{x}$ belongs to an unseen class $c \in \mathcal{C}^{\mathcal{U}}$. They propose to first estimate the class-conditional probability density $p(\mathbf{x}|c)$ for all seen classes $c \in \mathcal{C}^{\mathcal{S}}$. This is achieved by assuming the projections $\hat{\mathbf{s}}(\mathbf{x})$ of visual features in the semantic space[9] are normally distributed

[9] [138] actually corresponds to the CMT method from Section 1.2.2, so $\hat{\mathbf{s}}(\mathbf{x}) = \mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{x})$ (Equation (1.43)).

around the semantic prototype $\mathbf{s}_c$, so that

$$p(\mathbf{x}|c) = \mathcal{N}(\hat{\mathbf{s}}(\mathbf{x})|\mathbf{s}_c, \mathbf{\Sigma}_c),\ c \in \mathcal{C}^{\mathcal{S}} \tag{1.82}$$

where $\mathbf{\Sigma}_c$ is assumed to be diagonal and is estimated using the training samples.

It can then be considered that an instance $\mathbf{x}$ does not belong to a seen class if its class-conditional probability is below a threshold γ for all seen classes $c \in \mathcal{C}^{\mathcal{S}}$:

$$g_u(\mathbf{x}) = \mathbb{1}[\forall c \in \mathcal{C}^{\mathcal{S}},\ p(\mathbf{x}|c) < \gamma] \tag{1.83}$$

The threshold γ can be considered to be a hyper-parameter of the model.

Writing $g_s(\mathbf{x}) = 1 - g_u(\mathbf{x})$ the probability that $\mathbf{x}$ belongs to a seen class, it is also possible to use $g_s(\mathbf{x}) \propto \max_{c \in \mathcal{C}^{\mathcal{S}}} p(\mathbf{x}|c)$ so that estimated probabilities are not binary. The authors of [138] also propose further approaches for estimating $g_u(\mathbf{x})$, for example using unsupervised outlier detection approaches [71].

Provided the compatibility $f(\mathbf{x}, \mathbf{s}_c)$ can be interpreted as the probability that the label of visual instance $\mathbf{x}$ is c, i.e. $f(\mathbf{x}, \mathbf{s}_c) \approx P(y = c|\mathbf{x})$, the compatibilities of seen and unseen classes can be weighted by the estimated probabilities that $\mathbf{x}$ belongs to a seen or unseen class, so that

$$\hat{y} = \text{argmax} \left\{ f(\mathbf{x}, \mathbf{s}_c) g_u(\mathbf{x}) \right\}_{c \in \mathcal{C}^{\mathcal{U}}} \cup \left\{ f(\mathbf{x}, \mathbf{s}_c)(1 - g_u(\mathbf{x})) \right\}_{c \in \mathcal{C}^{\mathcal{S}}} \tag{1.84}$$

Alternatively [24] proposed to use a threshold γ (e.g. $\gamma = 0.5$) such that

$$\hat{y} = \begin{cases} \underset{c \in \mathcal{C}^{\mathcal{S}}}{\text{argmax}}\ f(\mathbf{x}, \mathbf{s}_c) & \text{if } g_u(\mathbf{x}) \leq \gamma \\ \underset{c \in \mathcal{C}^{\mathcal{U}}}{\text{argmax}}\ f(\mathbf{x}, \mathbf{s}_c) & \text{if } g_u(\mathbf{x}) > \gamma \end{cases} \tag{1.85}$$

For seen classes $c \in \mathcal{C}^{\mathcal{S}}$, $f(\mathbf{x}, \mathbf{s}_c)$ can be replaced by the output of a standard supervised classifier trained on seen classes.

Most recent GZSL methods adopt a more direct approach [152, 162, 23], refered to as *direct stacking* in [24]: the unweighted compatibility function f is used to directly estimate compatibilities of seen and unseen classes, so that we simply have

$$\hat{y} = \underset{c \in \mathcal{C}^{\mathcal{S}} \cup \mathcal{C}^{\mathcal{U}}}{\text{argmax}}\ f(\mathbf{x}, \mathbf{s}_c) \tag{1.86}$$

This approach has the advantage that using a trained ZSL model in a GZSL setting is straightforward, as all there is to do is adding the seen class prototypes to the list of prototypes whose compatibility with $\mathbf{x}$ needs to be evaluated. However, it has been empirically demonstrated that many ZSL models suffer from a bias towards seen classes when this approach is used to evaluate a classical ZSL model in GZSL setting [24, 163, 161]. More specifically, visual instances belonging to one of the unseen classes tend to be classified as instances from seen classes, resulting in a lower accuracy on samples from unseen classes than on sample from seen classes. As an example, a model trained with the seen classes *horse* and *fox* will tend to consider that a visual instance of the unseen class *zebra* is more likely a "weird" horse than a zebra when provided with the semantic prototypes of both classes. Some experimental results highlighting this effect are provided later in Table 2.6.

Measuring ZSL and GZSL score. The most obvious metric to measure the performance of a ZSL model is the standard accuracy, sometimes also called *per sample* accuracy, which is the micro-average rate of correct predictions. Given N test instances $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with ground-truth labels $\{y_1, \dots, y_N\}$ and associated predictions $\{\hat{y}_1, \dots, \hat{y}_N\}$, both belonging to a set of test classes $\mathcal{C}^{\text{te}}$, the per sample accuracy is

$$\mathcal{A}^{\text{p.s.}} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}[\hat{y}_n = y_n] = \frac{1}{N} \sum_{c \in \mathcal{C}^{\text{te}}} \sum_{m=1}^{M_c} \mathbb{1}[\hat{y}_m^c = y_m^c] \tag{1.87}$$

where $\hat{y}_m^c$ is the prediction corresponding to the m^{th} test sample of class c. In a *classical* ZSL setting – as opposed to a *generalized* ZSL setting – the test classes are all unseen classes, so that $\mathcal{C}^{\text{te}} = \mathcal{C}^{\mathcal{U}}$ (so that and $y_n, \hat{y}_n \in \mathcal{C}^{\mathcal{U}}$ for all n). On the other hand, in a GZSL setting, $\mathcal{C}^{\text{te}} = \mathcal{C} = \mathcal{C}^{\mathcal{S}} \cup \mathcal{C}^{\mathcal{U}}$.

Xian *et al.* [163] proposed to use *per class* accuracy to take class imbalance in certain benchmark datasets into account. Per class accuracy is the macro-average rate of correct predictions, i.e. the mean accuracy of the average accuracy $\frac{1}{M_c} \sum_{m=1}^{M_c} \mathbb{1}[\hat{y}_m^c = y_m^c]$ of each class c, each class having the same weight regardless of its number of associated test instances M_c:

$$\mathcal{A}^{\text{p.c.}} = \frac{1}{|\mathcal{C}^{\text{te}}|} \sum_{c \in \mathcal{C}^{\text{te}}} \frac{1}{M_c} \sum_{m=1}^{M_c} \mathbb{1}[\hat{y}_m^c = y_m^c] \tag{1.88}$$

Per class accuracy is much more common than per sample accuracy in the recent ZSL literature [163, 162, 152, 18, 28], so we will use $\mathcal{A} = \mathcal{A}^{\text{p.c.}}$ as the default measure of accuracy unless otherwise stated.

Notation	Description
$\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$	Accuracy on samples from unseen classes when candidate are unseen classes only
$\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$	Accuracy on samples from seen classes when candidate are seen classes only
$\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ or $\mathcal{A}_{\mathcal{U}}$	Accuracy on samples from unseen classes when candidate are seen and unseen classes
$\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ or $\mathcal{A}_{\mathcal{S}}$	Accuracy on samples from seen classes when candidate are seen and unseen classes
$\mathcal{A}_{\mathcal{C}\to\mathcal{C}}$ or $\mathcal{A}_{\mathcal{S}}$	Accuracy on samples from all classes when candidate are from all classes
$\mathcal{H}$	Harmonic mean of $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$

Table 1.5 – Summary of the notations used for accuracies on samples from seen and unseen classes. By default, we assume that per class accuracy is used.

However, in a GZSL setting, the accuracy alone does not always provide the full picture regarding the performance of a model: assuming per class accuracy is used and 80% of classes are seen classes, a perfect supervised model could achieve 80% accuracy with absolutely no ZSL abilities. This is all the more important as many GZSL approaches suffer from a bias towards seen classes as mentioned earlier.

To take the trade-off between performances on seen and unseen classes into account, accuracies on both types of class, seen and unseen, are often measured separately. Inspired by [24], $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ is defined in [163] as the (per class) accuracy evaluated on test instances of unseen classes when candidate classes are all classes $\mathcal{C}$, seen and unseen. Similarly, $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ is the accuracy evaluated on test instances of seen classes when candidate classes are all classes $\mathcal{C}$:

$$\mathcal{A}_{\mathcal{U}\to\mathcal{C}} = \frac{1}{|\mathcal{C}^{\mathcal{U}}|} \sum_{c\in\mathcal{C}^{\mathcal{U}}} \frac{1}{M_c} \sum_{m=1}^{M_c} \mathbb{1}[\hat{y}_m^c = y_m^c] \tag{1.89}$$

$$\mathcal{A}_{\mathcal{S}\to\mathcal{C}} = \frac{1}{|\mathcal{C}^{\mathcal{S}}|} \sum_{c\in\mathcal{C}^{\mathcal{S}}} \frac{1}{M_c} \sum_{m=1}^{M_c} \mathbb{1}[\hat{y}_m^c = y_m^c] \tag{1.90}$$

It is possible to similarly define $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ as the accuracy evaluated on unseen classes when candidate classes only consist in unseen classes, and $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ as the accuracy evaluated on seen classes when candidate classes only consist in seen classes. $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ corresponds to the classical ZSL accuracy defined in Equation (1.88) with $\mathcal{C}^{\text{te}} = \mathcal{C}^{\mathcal{U}}$. $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ corresponds to what is measured in a standard supervised learning setting. $\mathcal{A}_{\mathcal{C}\to\mathcal{C}}$ would correspond to the standard per class accuracy (Equation (1.88)) in

a GZSL setting, i.e. with $\mathcal{C}^{\text{te}} = \mathcal{C}^{\mathcal{S}} \cup \mathcal{C}^{\mathcal{U}}$. We will sometimes simply use $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ to refer to respectively $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$. This information is summarized in Table 1.5.

$\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ measure how well a GZSL model is performing on respectively seen and unseen classes. [163] proposed to use the harmonic mean $\mathcal{H}$ as a trade-off between the two measures, to avoid models with a very high score in one of these two sub-tasks but mediocre performance in the other. The final performance score of a GZSL is then

$$\mathcal{H} = \frac{2 \cdot \mathcal{A}_{\mathcal{U}} \cdot \mathcal{A}_{\mathcal{S}}}{\mathcal{A}_{\mathcal{U}} + \mathcal{A}_{\mathcal{S}}} \tag{1.91}$$

This measure is the most commonly used in the recent literature [161, 162, 152, 18, 28].

It can be noted that this metric requires to keep some instances from seen classes for the testing phase for a given ZSL benchmark dataset. For datasets where this is not convenient – for example if the number of training samples per class is really small or if the dataset suffers from biases (Appendix A.1), sometimes only $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ is evaluated in order to still provide some measure of GZSL performance [53].

Alternatively, Chao *et al.* [24] introduced *calibrated stacking*, where a weight γ is introduced in order to change the original prediction of the model based on whether the test sample belongs to a seen or an unseen class:

$$\hat{y} = \underset{c\in\mathcal{C}}{\text{argmax}}\ f(\mathbf{x}, \mathbf{s}_c) - \gamma \mathbb{1}[c \in \mathcal{C}^{\mathcal{S}}] \tag{1.92}$$

This weight can either favor $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ when $\gamma > 0$, or favor $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ when $\gamma < 0$. [24] defined the Area Under Seen-Unseen accuracy Curve (AUSUC) as the area under the curve of the plot with $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ on the x-axis and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ on the y-axis, when γ goes from $-\infty$ to $+\infty$. Similarly to the area under a Receiver Operating Characteristic (ROC) curve [38], the AUSUC can be used as a metric to evaluate the performance of a GZSL model. More details will be provided in Section 2.6.1

Chapter 2

Ranking methods and generalized zero-shot learning

Content

In this chapter, we focus on identifying aspects which arguably play an important role in zero-shot recognition, and which are not usually taken into account by existing zero-shot learning models. We more specifically focus on triplet loss methods (detailed in Section 1.2.3), and argue that such methods make several implicit hypotheses which may hinder their performance. We propose fairly simple modifications to existing approaches that may enable them to reach state-of-the-art performance, equalling or even surpassing the performance of generative models (Section 1.2.4) while being

more broadly applicable due to less restrictive hypotheses. We also focus on the bias benefiting the performance of seen classes in a generalized ZSL setting (Section 1.4) at the expense of unseen classes. We provide insights on why this bias appears and propose a simple solution resulting in important performance gains for all models and enabling our triplet loss approach to again equal or surpass the GZSL performance obtained with generative models.

This chapter is organized as follows: after a brief reminder on how triplet loss methods work, Sections 2.1, 2.2 and 2.3 argue that they usually make the implicit assumptions that respectively (1) classes are equally different, (2) the margin is an efficient regularizer, and (3) visual instances are all relevant. They also introduce the following respective mechanisms to overcome these limitations: (1) a flexible semantic margin, (2) a partial normalization and (3) a sample relevance weighting scheme. Section 2.4 combines these contributions in a cohesive ZSL model, whose performance is evaluated in Section 2.5. Section 2.5 also highlights the gap in performance between the accuracies on seen and unseen classes in a GZSL setting. In Section 2.6, we attempt to explain why this bias exists and propose a simple mechanism to hinder its impact, whose effectiveness is finally evaluated in Section 2.7. Section 2.8 provides a discussion on some of the design choices made in this chapter. Throughout this chapter, we use the same notations as in sections 1.1.2 and 1.2. Such notations are summarized in Table 1.2. Many concepts in this chapter are illustrated with examples from the CUB dataset [154], which is described in Appendix A.1.

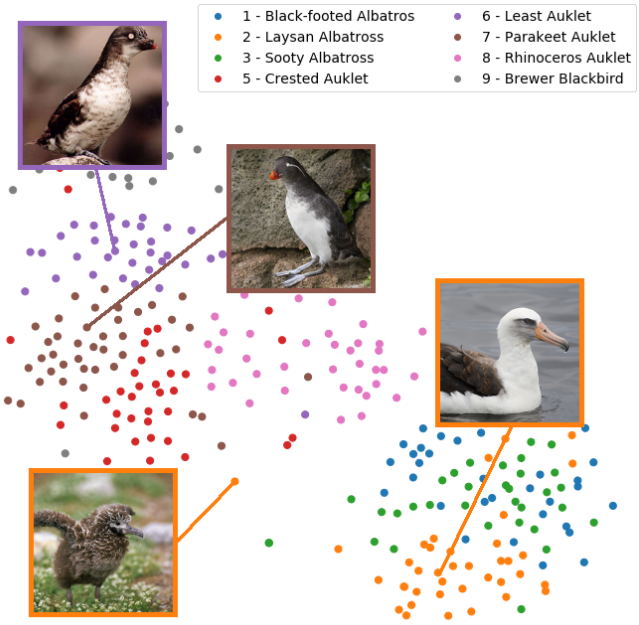

Figure 2.1 – t-SNE [96] visualization of 300 visual instances from the first 8 training classes of the CUB dataset. Classes *least auklet* (purple) and *parakeet auklet* (brown) are much more similar to each other than classes *least auklet* and *laysan albatross* (orange). The nestling from class *laysan albatross* is quite dissimilar from other samples from this class.

2.1 Semantic margin

As explained earlier in Section 1.2.3, triplet loss methods are based on the intuition that each visual sample should be "much" more compatible with the prototype corresponding to its class than to all the others given a compatibility function f. More formally, they aim to enforce $f(\mathbf{x}, \mathbf{s}_y) \gg f(\mathbf{x}, \mathbf{s}_c)$, with $(\mathbf{x}, y)$ a labeled visual sample, $\mathbf{s}_y$ the corresponding semantic prototype and $\mathbf{s}_c$ another prototype $(c \neq y)$. This is achieved by enforcing

$$f(\mathbf{x}, \mathbf{s}_y) \geq m + f(\mathbf{x}, \mathbf{s}_c) \tag{2.1}$$

for a fixed margin m. This can be expressed with the triplet loss

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{s}_c, \mathbf{s}_y; f) = [m + f(\mathbf{x}, \mathbf{s}_c) - f(\mathbf{x}, \mathbf{s}_y)]_+ \tag{2.2}$$

It can be noticed that the margin is not strictly necessary, as it should be enough that $f(\mathbf{x}, \mathbf{s}_y) > f(\mathbf{x}, \mathbf{s}_c)$. But similarly to SVMs, the margin plays a regularization role, as this approach is inspired

by and is a generalization of the hinge loss as explained in Section 1.2.3.

A straightforward implementation of a triplet loss, such as DeViSE [40], consists in using a bilinear compatibility function $f_{\mathbf{W}}(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \mathbf{W}\mathbf{s}$ and simply summing the loss from Equation (2.2) over all combinations of training samples and prototypes:

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{\substack{c \in \mathcal{C}^\mathcal{S} \\ c \neq y_n}} [m + f(\mathbf{x}_n, \mathbf{s}_c) - f(\mathbf{x}_n, \mathbf{s}_{y_n})]_+ \tag{2.3}$$

Several other variations of this idea [4, 3] have been detailed in the previous section. However, although these methods have led to promising results for ZSL, we argue that they fail to consider several important aspects of the problem due to implicit hypotheses.

The first such hypothesis is that *classes are equally different*, as there is no difference between any two incorrect class assignments in the triplet loss in Equation (2.2). However, in many cases, and particularly in fine-grained datasets comprising many classes, there may be groups of very similar classes. Figure 2.1 illustrates such a case: two samples from classes 6 *"least auklet"* and 7 *"parakeet auklet"* from the CUB dataset are much more difficult to tell apart than two samples from classes 6 *"least auklet"* and 2 *"laysan albatross"*. These classes *"least auklet"* and *"parakeet auklet"* may be hard to tell apart even for standard supervised models, and it may be impossible not to violate inequality 2.1. When building the similarity-based decision model, a confusion between two nearly indistinguishable classes should arguably not be penalized as much as a confusion between two grossly different classes.

To improve the robustness of the learned mapping between the semantic and visual spaces, we propose to replace the fixed margin m in equations (2.1) and (2.2) by a variable margin $m(c, c')$ measuring the (dis)similarity between classes c and c'. This way, for very similar classes c and c' with dissimilarity close to 0, it is enough that $f(\mathbf{x}, \mathbf{s}_c) > f(\mathbf{x}, \mathbf{s}_{c'})$, c being the class corresponding to $\mathbf{x}$. Conversely, very dissimilar classes should be easy to tell apart, and we expect $f(\mathbf{x}, \mathbf{s}_c) > M + f(\mathbf{x}, \mathbf{s}_{c'})$, with $M = m(c, c')$ being very large.

We propose to measure this similarity in the semantic space. As attributes tend to be correlated [59], we use a Mahalanobis distance to take these correlations into account. The dissimilarity between classes i and j with respective prototypes $\mathbf{s}_i$ and $\mathbf{s}_j$ is therefore

$$m(i, j) = \Big((\mathbf{s}_i - \mathbf{s}_j)\mathbf{\Sigma}^{-1}(\mathbf{s}_i - \mathbf{s}_j)\Big)^{\frac{1}{2}} \tag{2.4}$$

where $\mathbf{\Sigma}^{-1}$ is the inverse of the covariance matrix between attributes, which can be estimated using the prototypes of seen classes.

The impact of using a Mahalanobis distance as opposed to a standard euclidean distance will be measured in Section 2.5 and are available in Table 2.2.

One problem with the Mahalanobis distance is that its estimation does not scale well with the dimension K of the semantic space, as $\mathcal{O}(K^2)$ parameters need to be estimated. For the CUB dataset with 312 attributes, this represents 156×313 parameters, which are learned using the prototypes of only 150 seen classes. We therefore need to employ methods which enable a robust estimation of the (inverse of) the covariance matrix. An example of such a method is the Ledoit-Wolf method [87], which produces a regularized version of the maximum likelihood estimation of the covariance matrix. We use the corresponding scikit-learn [116] implementation in our experiments.

We are still left with a few issues. First, the expected (mean) distance between two classes $m(c, c')$ in the semantic space is arbitrary, and is not necessarily a suitable value for the margin. Second, since the semantic space is usually high-dimensional, these distances typically have low variance, so that distances are close to the mean. This means that the margins corresponding to two very similar and two very dissimilar classes will not be significantly different, and therefore will not have a large impact during the training of the model. Both of these aspects are illustrated in Figure 2.2 (left).

As a solution, we propose to rescale the distances to have a mean $\mu_{\mathbf{M}}$ and a standard deviation $\sigma_{\mathbf{M}}$ set by the user. We write $\mathbf{M} \in \mathbb{R}^{K \times K}$ the matrix representing the distances between all pairs of classes, such that $M_{i,j} = m(i, j)$. We estimate the empirical mean $\hat{\mu}_{\mathbf{M}}$ and standard deviation $\hat{\sigma}_{\mathbf{M}}$ of these distances:

$$\hat{\mu}_{\mathbf{M}} = \frac{1}{K^2} \sum_{i=1}^{K} \sum_{j=1}^{K} M_{i,j} \tag{2.5}$$

$$\hat{\sigma}_{\mathbf{M}} = \sqrt{\frac{1}{K^2} \sum_{i=1}^{K} \sum_{j=1}^{K} (1 - \hat{\mu}_{\mathbf{M}})^2} \tag{2.6}$$

We can then rescale the elements of $\mathbf{M}$ so that they have approximately mean $\mu_{\mathbf{M}}$ and standard deviation $\sigma_{\mathbf{M}}$ while still being positive:

$$\mathbf{M} = \left[\frac{\mathbf{M} - \hat{\mu}_{\mathbf{M}}}{\hat{\sigma}_{\mathbf{M}}} \sigma_{\mathbf{M}} + \mu_{\mathbf{M}} \right]_+ \tag{2.7}$$

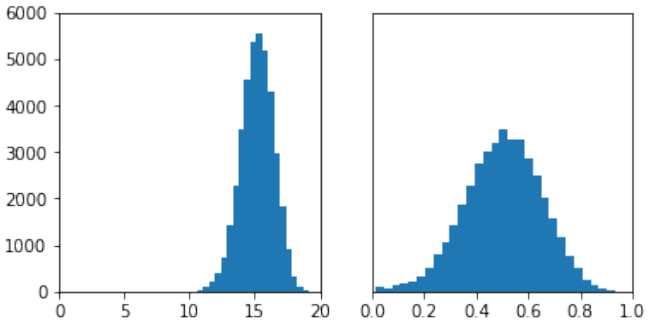

Figure 2.2 – *Left*: histogram of the raw semantic distances $\mathbf{M}$ as measured on the seen classes from the CUB dataset, with mean distance $\hat{\mu}_{\mathbf{M}}$ and standard deviation $\hat{\sigma}_{\mathbf{M}}$ approximately equal to 15.4 and 1.2. *Right*: rescaled with $\mu_{\mathbf{M}}$ and $\sigma_{\mathbf{M}}$ set to respectively 0.5 and 0.15.

The elements of this new matrix $\mathbf{M}$ correspond to the final margins used between all pairs of classes: for the two classes i and j corresponding to row i and column j of $\mathbf{M}$, we use the margin $m(i, j)$ such that

$$m(i, j) = M_{i,j} \tag{2.8}$$

We consider that $\mu_{\mathbf{M}}$ and $\sigma_{\mathbf{M}}$ are hyperparameters of the model. An illustration of the rescaled distances with set mean $\mu_{\mathbf{M}}$ and standard deviation $\sigma_{\mathbf{M}}$ is shown in Figure 2.2 (right). We can note that with $\sigma_{\mathbf{M}}$ close to 0 and $\mu_{\mathbf{M}}$ set to 1, for example, the semantic margin is no longer variable and the method is equivalent to DeViSE.

Illustrations of the most similar and least similar classes to classes *"red-legged kittiwake"* and *"arctic tern"* from the CUB dataset are available in Figure 2.3; additional illustrations are provided in Appendix A.3. Interestingly, some form of the hubness phenomenon can be observed, as *"American crow"* and *"fish crow"* seem to be among the most similar classes of many classes. The most and least similar classes otherwise appear to be fairly consistent.

Experimental results including the use of the variable semantic margin are provided in Section 2.5.

2.2 Impact of the margin

In equations (2.1) and (2.2), the margin m is supposed to act as a regularizer and reduce overfitting on the training set. We intuitively expect that a larger value of the margin m should incentivize the model to increase the difference $f(\mathbf{x}, \mathbf{s}_y) - f(\mathbf{x}, \mathbf{s}_c)$ between compatibility of matching pair $f(\mathbf{x}, \mathbf{s}_y)$ and

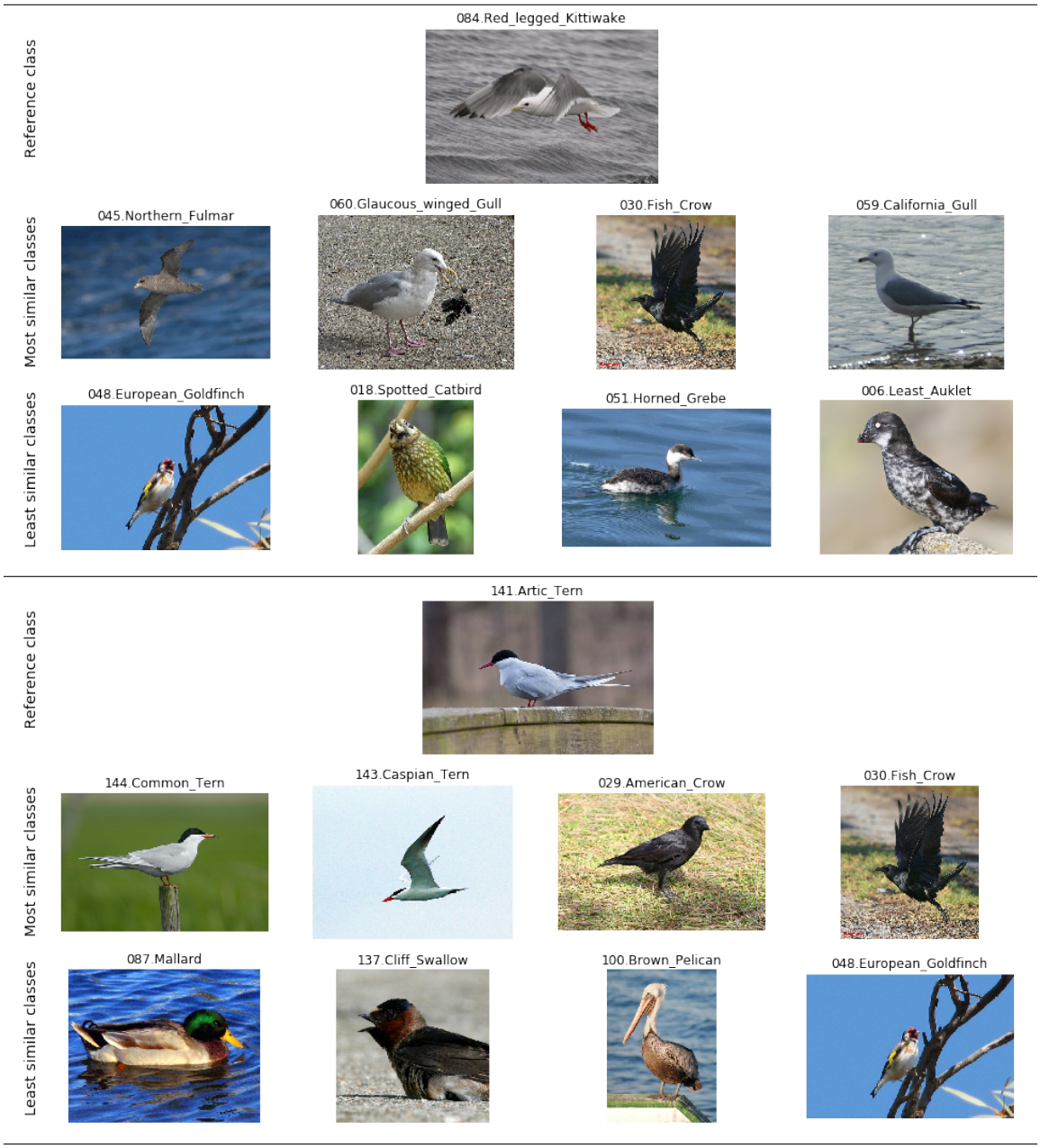

Figure 2.3 – Most similar and least similar classes to classes *"red-legged kittiwake" (top)* and *"arctic tern" (bottom)* from the CUB dataset, as measured by Equation (2.4). Examples for additional classes are provided in Appendix A.3.

compatibility of non matching pair $f(\mathbf{x}, \mathbf{s}_c)$, and thus improve the robustness of the model – up to the point where the constraint in Equation (2.1) becomes unsatisfiable. However, this is not always what happens, particularly with dot product compatibility functions. This is what we call the *assumption that the margin is an efficient regularizer*. We will discuss both theoretical and experimental aspects.

As mentioned in Section 1.2.3, the compatibility function f often takes the form of a dot product between the projection $\boldsymbol{\theta}(\mathbf{x})$ of a visual sample $\mathbf{x}$ and the projection $\boldsymbol{\phi}(\mathbf{s})$ of a semantic prototype $\mathbf{s}$:

$$f(\mathbf{x}, \mathbf{s}) = \boldsymbol{\theta}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{s}) \tag{2.9}$$

For example, for models with a linear compatibility function such as DeViSE, ALE or SJE from Section 1.2.3, we have $\boldsymbol{\theta}(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ and $\boldsymbol{\phi}(\mathbf{s}) = \mathbf{s}$, which results in the bilinear compatibility function $f(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \mathbf{W} \mathbf{s}$ from Equation (1.48). This dot product $\boldsymbol{\theta}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{s})$ can also be written

$$\boldsymbol{\theta}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{s}) = \|\boldsymbol{\theta}(\mathbf{x})\|_2 \cdot \|\boldsymbol{\phi}(\mathbf{s})\|_2 \cdot \cos(\alpha) \tag{2.10}$$

where α is the angle between $\boldsymbol{\theta}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{s})$.

Thus, the value of the compatibility $f(\mathbf{x}, \mathbf{s})$ depends on three components: the norm $\|\boldsymbol{\theta}(\mathbf{x})\|_2$ of the projected visual sample, the norm $\|\boldsymbol{\phi}(\mathbf{s})\|_2$ of the projected prototype, and the cosine $\cos(\alpha)$ of the angle between the projections $\boldsymbol{\theta}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{s})$. The cosine of the angle α is obviously bounded. Since $\mathbf{s}$ is usually unit-normalized, so is $\boldsymbol{\phi}(\mathbf{s})$ when $\boldsymbol{\phi}$ is the identity. However, the norm of $\boldsymbol{\theta}(\mathbf{x})$ is not bounded.

In practice, this means that if we double the margin m with respect to a base model, the new model can simply double the difference $f(\mathbf{x}, \mathbf{s}_y) - f(\mathbf{x}, \mathbf{s}_c)$ in Equation (2.2) by doubling the compatibility $f(\mathbf{x}, \mathbf{s})$ for all $\mathbf{s}$, which can be achieved by simply doubling the norm $\|\boldsymbol{\theta}(\mathbf{x})\|_2$ of the projection $\boldsymbol{\theta}(\mathbf{x})$. This can result in a new model very similar to the base model despite a different value of m, where the values of $\boldsymbol{\theta}(\mathbf{x})$ have simply been doubled. Therefore, the actual value of the margin m has little impact on the parameters learned by the model.

This effect can be empirically measured. The blue line (corresponding to $\rho = 0$) in Figure 2.4 represents the average norm of $\boldsymbol{\theta}(\mathbf{x})$ as measured on the training samples of the CUB dataset. Triplet loss models similar to DeViSE are trained on the dataset, with values of the margin m ranging from 0.2 to 2.0. It can be seen that the average norm of the projected visual features does indeed increase significantly with m.

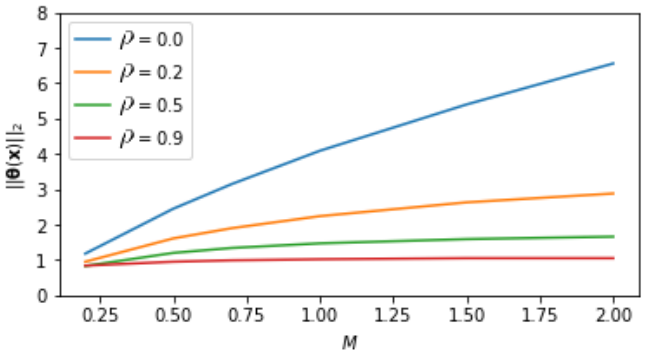

Figure 2.4 – Average norm of the projected visual features $\|\boldsymbol{\theta}(\mathbf{x})\|_2$ with respect to the margin M as measured on the CUB dataset. The value $\rho = 0$ corresponds to no (partial) normalization.

This makes the actual value of the margin m of little relevance, and thus reduces the regularization provided by the margin.

Several solutions can be considered to address this problem: (1) Regularize the parameters of $\boldsymbol{\theta}(\cdot)$ with an $\ell 2$ penalty to mitigate the scaling of the norm of $\boldsymbol{\theta}(\mathbf{x})$ with m; or (2) Fully normalize $\boldsymbol{\theta}(\mathbf{x})$ so that it always has unit norm. However, none of these solutions led to satisfying results in our preliminary experiments. In particular, completely removing the constraint of $\boldsymbol{\theta}(\mathbf{x})$ having consistent norms – when $\boldsymbol{\theta}(\mathbf{x})$ is fully normalized – led to severe overfitting on the some datasets.

We thus a introduce a *partial normalization* function $\boldsymbol{\Psi}$, taking a vector $\mathbf{v}$ as input and parameterized by a scalar $\rho \in [0, 1]$ such that

$$\boldsymbol{\Psi}_\rho(\mathbf{v}) = \frac{1}{\rho\,(\|\mathbf{v}\|_2 - 1) + 1} \cdot \mathbf{v} \tag{2.11}$$

A value of $\rho = 0$ means no transformation is applied to $\mathbf{v}$, and a value of $\rho = 1$ means that $\mathbf{v}$ is fully normalized to have unit euclidean norm. Values between 0 and 1 have intermediate results.

The value of ρ is then considered to be a hyperparameter of the model, which controls how much we allow the model to scale the projections of visual samples to deal with a large margin. Figure 2.4 shows the effect of different values of ρ regarding the scaling of the projected norm of $\boldsymbol{\theta}(\mathbf{x})$ with respect to the margin m as measured on the CUB dataset[1]. It is important to realize that when $\rho \neq 1$, $\boldsymbol{\theta}(\mathbf{x})$

[1]It is interesting to observe that on this specific dataset, cross-validation leads to selecting values of ρ close to 1, and results obtained with fully normalized projections are unsurprisingly very close. The results are not as pronounced on other datasets such as AwA2 [161].

can still be scaled up to compensate for the effect of the partial normalization, which therefore needs to be combined with a regularization on $\boldsymbol{\theta}(\mathbf{x})$ – e.g. an $\ell 1$ or $\ell 2$ penalty – to be fully effective.

2.3 Relevance weighting

The third assumption made by triplet loss methods, and more broadly by ZSL methods, is that all training samples are relevant. This means that all the samples from seen classes are considered equally representative when building the model. However, as illustrated in Figure 2.1, some samples can be quite different from most samples from their class, sometimes to the point of being nearly impossible to correctly classify even in a standard supervised learning setting. This can have an adverse effect during training if these atypical samples contribute a lot to the training loss, encouraging the model to focus specifically on these samples – possibly at the expense of more typical samples.

We propose to assign a score to each training sample to quantify its "representativeness" with respect to its class, and to weigh the loss associated with each training sample using this score. This way, a mistake regarding a sample deemed typical should be way more penalized than a mistake regarding an outlier during training, which should improve the robustness of the learned multi-modal relations.

For each class c, we compute the mean visual representation $\overset{*}{\mathbf{x}}_c$. For each sample $\mathbf{x}_m^c$ belonging to this class, we then compute the distance u_m^c to the mean visual representation of class c in the visual space.

$$\overset{*}{\mathbf{x}}_c = \frac{1}{M_c} \sum_{m=1}^{M_c} \mathbf{x}_m^c \tag{2.12}$$

$$u_m^c = \|\mathbf{x}_m^c - \overset{*}{\mathbf{x}}_c\|_2 \tag{2.13}$$

Provided the visual features are suitable for these distances to be meaningful, this provides a first estimation of how different an image is from the other images in the same class.

However, similarly to the semantic distances in Section 2.1, the scale of these distances is somehow arbitrary. In particular, this may be problematic when classes have different intra-class variance: classes whose instances are on average farther away from the center than other classes will be assigned

smaller weights, and will contribute less to the training loss.

We therefore normalize these distances so that they are roughly on the same scale regardless of the inter-class variance. We compute the empirical mean μ_c and standard deviation σ_c of the distances from Equation (2.13) for each class c, and set the distances u_m^c so that they have zero mean and unit variance for all classes.

$$\mu_c = \frac{1}{M_c} \sum_m u_m^c \tag{2.14}$$

$$\sigma_c = \sqrt{\frac{1}{M_c} \sum_m (u_m^c - \mu_c)^2} : \tag{2.15}$$

$$u_m^c = (u_m^c - \mu_c)/\sigma_c \tag{2.16}$$

We finally define the relevance weights v_m^c based on the rescaled distances u_m^c so that larger weights mean "more relevant" samples and such that they belong to the interval $[0, 1]$:

$$v_m^c = 1 - \Phi\left(u_m^c\right) \tag{2.17}$$

where $\Phi(\cdot)$ is the cumulative density function of the normal distribution[2]. This way, instances which are very far away from the centroid of their class have a weight close to 0, and instances very close to the centroid have a weight close to 1.

Although this is not made explicit here, there is obviously a one-to-one correspondence between the n^{th} sample $\mathbf{x}_n$ of the whole dataset, with label y_n, and the m^{th} sample $\mathbf{x}_m^{y_n}$ of class y_n for some m. The weight $v_m^{y_n}$ associated with this $\mathbf{x}_m^{y_n}$ is thus also the weight associated with $\mathbf{x}_n$. We will simply write v_n to refer to the relevance weight corresponding to the n^{th} instance $\mathbf{x}_n$.

The distribution of all the weights v_m^c obtained with Equation (2.17) for the class *"laysan albatross"* from the CUB dataset is illustrated in Figure 2.5. Illustrations of the most similar and least relevant instances for classes *"red-legged kittiwake"* and *"arctic tern"* from the CUB dataset are available in Figure 2.6; additional illustrations are provided in Appendix A.3.

It is important to note that while this relevance weighting scheme can improve the overall robustness of the model (see experiments in Section 2.5) by relating the impact of each training sample to its

[2]This choice is motivated by the fact that the distribution of the weights roughly has a Gaussian shape but is somewhat arbitrary, and very similar results can be obtained with a sigmoid function.

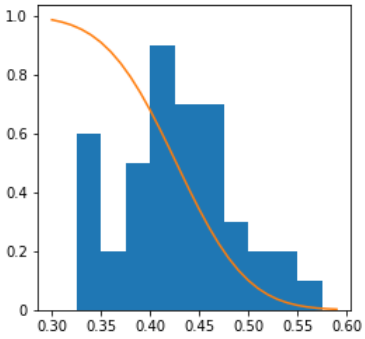

Figure 2.5 – Histogram *(blue)* of the distances to the class center $\overset{*}{\mathbf{x}}_c$ (Equation (2.12)) and associated weights *(orange)* from Equation (2.17) for visual samples from class *"laysan albatross"* from the CUB dataset. The weights of the nestling and adult samples represented in Figure 2.1 are respectively 0.02 and 0.78.

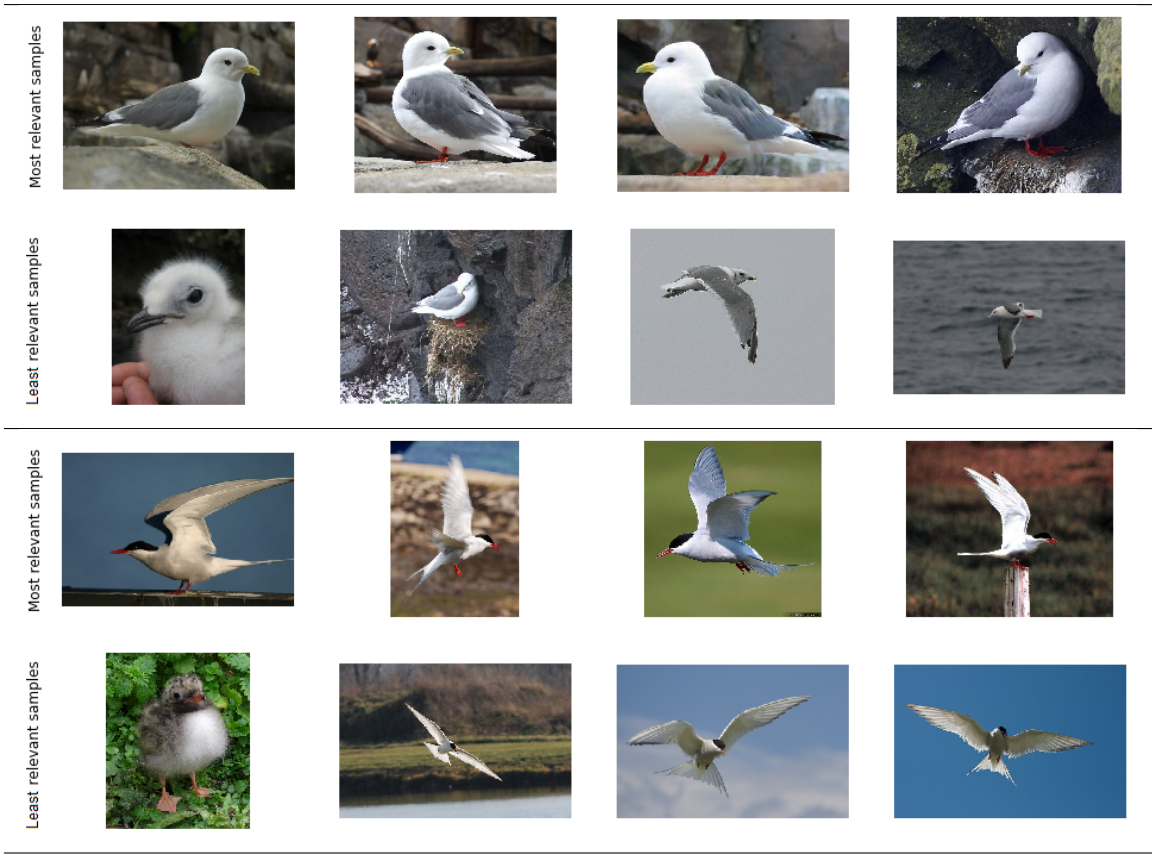

Figure 2.6 – Most and least representative samples from classes *"red-legged kittiwake" (top)* and *"arctic tern" (bottom)* from the CUB dataset, as measured by Equation (2.17). Examples for additional classes are provided in Appendix A.3.

representativeness of the class, it also makes atypical samples harder to recognize. With the example of the CUB dataset, nestlings are treated as outliers because they are significantly under-represented, but the corresponding images still belong to their respective species. Being unable to recognize atypical samples may be an issue for some practical cases. A way to circumvent the problem could be to define sub-classes and process each of them separately, provided that each is sufficiently well represented in the training set. For example, on the seen classes, sub-classes could be found by applying clustering algorithms to the visual features. However, usual ZSL benchmarks such as CUB often do not have enough samples of such sub-categories for these types of approaches to be meaningful – at most a single instance of a nestling is present for most classes in CUB as illustrated in Table 2.6. As a result, we do not explore these approaches.

2.4 Proposed model

We can now bring together the ideas from sections 2.1, 2.2 and 2.3 into a single ZSL triplet loss model which makes use of a flexible semantic margin, partial normalization and relevance weighting.

The model consists in learning projections $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ which respectively map visual samples $\mathbf{x}$ and semantic prototypes $\mathbf{s}$ to a common space, such that the compatibility between $\mathbf{x}$ and $\mathbf{s}$ can be evaluated in this space with a dot product. Partial normalization $\boldsymbol{\Psi}_\rho$ (Equation (2.11)) is applied to the resulting projections $\boldsymbol{\theta}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{s})$ before performing the dot product. We therefore have a compatibility function f such that

$$f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_n, \mathbf{s}_c) = \boldsymbol{\Psi}_\rho(\boldsymbol{\theta}(\mathbf{x}_n))^\top \boldsymbol{\Psi}_\rho(\boldsymbol{\phi}(\mathbf{s}_c)) \tag{2.18}$$

Using $m(c, c')$ from Equation (2.8) as the flexible semantic margin between two classes c and c', for a triplet $(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c)$, $c \neq y_n$, the triplet loss now takes the form

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c; f_{\boldsymbol{\theta},\boldsymbol{\phi}}) = [m(y_n, c) + f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_n, \mathbf{s}_c) - f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_n, \mathbf{s}_{y_n})]_+ \tag{2.19}$$

We adopt the simplest approach to apply this triplet loss to the training set: it consists in simply summing this triplet loss over all training triplets $(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c)$, $c \neq y_n$, in a setting similar to DeViSE. The loss from each triplet is further weighted by the weight v_n from Equation (2.17). The resulting

final loss function is:

$$\frac{1}{N}\sum_{n=1}^{N}\left(v_n \sum_{\substack{c\in\mathcal{C}^{\mathcal{S}}\\ c\neq y_n}} \mathcal{L}_{\text{triplet}}(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c; f_{\boldsymbol{\theta},\boldsymbol{\phi}})\right) + \lambda\Omega[f_{\boldsymbol{\theta},\boldsymbol{\phi}}] \tag{2.20}$$

where λ is a hyperparameter controlling the weight of the regularization Ω. $\Omega[f_{\boldsymbol{\theta},\boldsymbol{\phi}}]$ is defined as the sum of the normalized $\ell 1$ norms[3] of the respective parameters $\theta_1, \ldots, \theta_P$ and $\phi_1, \ldots, \phi_Q$ of projections $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$:

$$\Omega[f_{\boldsymbol{\theta},\boldsymbol{\phi}}] = \frac{1}{P}\sum_{p=1}^{P}|\theta_p| + \frac{1}{Q}\sum_{q=1}^{Q}|\phi_q| \tag{2.21}$$

In our implementation, we choose simple linear projections for $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. We actually consider *two variants* of the model: in the first one, we only project the visual features $\mathbf{x}$ onto the semantic space, so that

$$\boldsymbol{\theta}(\mathbf{x}) = \mathbf{W}_{\boldsymbol{\theta}} \cdot \mathbf{x} \tag{2.22}$$

$$\boldsymbol{\phi}(\mathbf{s}) = \mathbf{s} \tag{2.23}$$

with $\mathbf{W}_{\boldsymbol{\theta}} \in \mathbb{R}^{K\times D}$. This corresponds to the setting from DeViSE, save the partial normalization (and relevance weight and semantic margin).

In the second variant, we linearly project both $\mathbf{x}$ and $\mathbf{s}$ onto a common space with the same dimension K as the semantic space, so that

$$\boldsymbol{\theta}(\mathbf{x}) = \mathbf{W}_{\boldsymbol{\theta}} \cdot \mathbf{x} \tag{2.24}$$

$$\boldsymbol{\phi}(\mathbf{s}) = \mathbf{W}_{\boldsymbol{\phi}} \cdot \mathbf{s} \tag{2.25}$$

with $\mathbf{W}_{\boldsymbol{\theta}} \in \mathbb{R}^{K\times D}$ and $\mathbf{W}_{\boldsymbol{\phi}} \in \mathbb{R}^{K\times K}$. Both variants $\boldsymbol{\theta} + \mathbf{I}$ (equations (2.22) and (2.23)) and $\boldsymbol{\theta} + \boldsymbol{\phi}$ (equations (2.24) and (2.25)) can either be evaluated separately, or we can consider that the variant itself is a hyperparameter and choose the variant with the best score on the validation set.

To simplify the model and since semantic prototypes are frequently $\ell 2$-normalized [24, 22], we choose to always fully normalize the projection of $\mathbf{s}_c$, so that its final projection is $\boldsymbol{\Psi}_1(\boldsymbol{\phi}(\mathbf{s}_c)) = \frac{\boldsymbol{\phi}(\mathbf{s}_c)}{\|\boldsymbol{\phi}(\mathbf{s}_c)\|_2}$ (and $\mathbf{s}_c$ is still normalized when $\boldsymbol{\phi}(\mathbf{s}) = \mathbf{s}$).

[3]Similarly to [22], an $\ell 1$ regularization can introduce some sparsity so that only a subset of attributes is used for instance.

We employ the following protocol to select the hyper-parameters $\mu_{\mathbf{M}}$ and $\sigma_{\mathbf{M}}$ (Equation (2.7)), ρ (Equation (2.11)), and λ (Equation (2.20)): for a given variant, whether $\boldsymbol{\theta} + \mathbf{I}$ (equations (2.22) and (2.23)) or $\boldsymbol{\theta} + \boldsymbol{\phi}$ (equations (2.24) and (2.25)), we first set $\sigma_{\mathbf{M}}$ and ρ to 0, and $\mu_{\mathbf{M}}$ to 1, so that the setting is approximately the one from DeViSE. We determine the best λ using the validation set(s). We then divide this value by a factor of 10 in order to not over-constrain the model, and use this new value when jointly selecting ρ and $\mu_{\mathbf{M}}$. We select $\sigma_{\mathbf{M}}$ while keeping the other hyper-parameters fixed. Finally, we explore values in the neighborhood of the selected quadruplet $(\mu_{\mathbf{M}}, \sigma_{\mathbf{M}}, \rho, \lambda)$. We retain the variant of the model producing the best results on the validation set.

Although this framework does introduce a number of hyper-parameters which may seen cumbersome to select, the whole framework can be seen as being simply a generalization of DeViSE or other simple triplet loss models, and components can be easily activated or deactivated depending on the specificities of the dataset on which it is applied. We have already seen that setting $\sigma_{\mathbf{M}}$ close to 0 and $\mu_{\mathbf{M}}$ close to 1 removes the effect of the flexible semantic margin, which may not be needed if the dataset only consist of fairly dissimilar classes. Similarly, setting $\rho = 0$ deactivates the partial normalization. In some cases, it may also be possible to set $\rho = 1$, in which case the regularization Ω may not be needed. Setting $v_n = 1$ for all n in Equation (2.20) deactivates relevance weighting. Using the $\boldsymbol{\theta} + \mathbf{I}$ variant and further fixing $\lambda = 0$ in Equation (2.20) yields a model identical to DeViSE.

One advantage of this approach is that whatever the training setting, we simply learn a (bi)linear compatibility function. Once the model is trained, making predictions is as easy as for any model with a direct compatibility function: given a test sample $\mathbf{x}$, we predict

$$\hat{y} = \underset{c \in \mathcal{C}^U}{\operatorname{argmax}} \; f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, \mathbf{s}_c) \tag{2.26}$$

using the formulation of $f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, \mathbf{s}_c)$ from Equation (2.18).

2.5 Experimental evaluation of the proposed method

We now evaluate the effectiveness of the proposed approach. The experimental protocol, detailed below, is essentially the same as in [161], which provides a fair comparison of many ZSL models under similar experimental settings and has become a reference benchmark in recent literature [161, 162, 152, 18, 28]. In particular, we use the same metrics to measure performance: the per-class accuracy $\mathcal{A}^{\text{p.c.}}$ or $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ for the ZSL setting, and the harmonic mean $\mathcal{H}$ of $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ for the GZSL setting, as

detailed in Section 1.4. We compare our results to those reported in [161]. We further independently reproduce some of these results later in Section 2.7. We also report results obtained with additional relevant ZSL models as detailed below.

Datasets and splits. Similarly to [161], we perform evaluation on three standard ZSL benchmarks[4]: the Caltech UCSD Birds 200-2011 dataset (**CUB**) [154], the SUN Attribute dataset (**SUN**) [114]) and the Animals with Attributes 2 dataset (**AwA2**) [161]. CUN and SUN are fine-grained datasets representing 200 bird species for the former and 717 scenes for the latter, while AwA2 is a more coarse-grained dataset of 50 animal species. More details on each dataset are provided in Appendix A.1.

We use the "Proposed Splits" introduced in [163] to avoid biases resulting from the presence of unseen classes in the ImageNet [33] dataset used to pre-train the visual features extractor, as detailed in Section 1.3.1. Also similarly to [163], we randomly remove 20% of the images from seen classes, which we use as unseen samples from seen classes during testing in the GZSL setting.

We select the hyper-parameters with 3-fold cross-validation on the training sets for CUB and SUN, by using some of the seen classes as unseen validation classes as illustrated later in Figure 2.8 (top). More details on cross-validation for (G)ZSL will be provided in Section 2.6.2. For AwA2, the selection of the validation set is more difficult, as among the 40 training classes only 8 are not in ImageNet. As a consequence, randomly selected cross-validation folds would contain few such classes. This may introduce significant differences between hyper-parameter values that are optimal for cross-validation folds (as they would mostly contain ImageNet classes) and those optimal for truly unseen classes. We therefore use a single validation split containing all 8 classes that are not in ImageNet, and perform each evaluation of a set of hyper-parameters with 3 different random initializations of parameters in order to improve the robustness of the estimate.

[4]In [161], in addition to CUB, SUN and AwA2, experiments are also conducted on the Attributes Pascal and Yahoo dataset (aPY) [37] and on the Animal with Attributes dataset (AwA1) [75]. However, AwA1 is not available anymore due to copyright issues, so we could not measure the performance of our approach on this dataset. Instead, AwA2 is used as a replacement. As for aPY, as detailed in Appendix A.1, this dataset suffers from very large class imbalance and biases, so we choose not to use it since evaluation on this dataset is not common in recent literature [162, 152, 164, 23]. [161] also conducts large-scale experiments on the ImageNet [33] dataset. However, results in the setting of [161] cannot be fairly compared since [53] demonstrated that this setting induces a large bias. Large scale ZSL will be discussed in depth in Chapter 3.

Visual and semantic features. We employ a ResNet-101 [55] network pre-trained on ImageNet as a deep visual feature extractor in order to have results comparable with the rest of the state-of-the-art, and in particular with [163] in which similar visual features are used. Keeping the activation weights of the last pooling layer gives us a 2048-dimensional visual feature representation as detailed in Section 1.3.1. Because we need a robust representation to compute distances between visual samples, we apply "10-crop" on the original images (Section 1.3.1), i.e. each 256×256 image is cropped into ten 224×224 images: one in each corner and one in the center for both the original image and its y-axis symmetry. The visual features of the resulting images are then averaged to obtain a 2048-dimensional vector. The visual vectors are finally normalized so that each vector of visual features $\mathbf{x}$ has unit $\ell 2$ norm.

We employ the standard attributes provided with each ZSL dataset, which we normalize to obtain class prototypes having unit $\ell 2$ norm.

Zero-shot learning models. We compare our approach to a number of ZSL approaches. We report results for the most relevant and best performing models from [161]: we include results for DeViSE [40], SJE [4] and ALE [2] as similarly to our approach, they are models based on a triplet loss and a bilinear compatibility function. We also include SynC [22] and GFZSL [153], which are frequently the best performing models in [161]. To have models based on different approaches, we also include results for the ridge regression approaches ESZSL [130] and SAE [70] from [161], and independently implement the $\text{Ridge}_{\mathcal{V}\rightarrow\mathcal{S}}$and $\text{Ridge}_{\mathcal{S}\rightarrow\mathcal{V}}$models (Section 1.2.2). Finally, we include results reported in other relevant works, for instance the PSR [5] approach, which considers how close classes are in the attribute space by explicitly including this information in the objective function to learn a mapping from the attribute space to the visual space.

We also include results from a few state-of-the art generative approaches such as SE-GZSL [152], f-GAN [162] and GMMM-ZSL [17, 18]. However, as mentioned in Section 1.4, the generative models trained in a class-inductive setting usually require additional training or processing steps to generate samples and train a supervised classifier given the prototypes of unseen classes, so integrating a single class into the model is not as straightforward as with other approaches. As a result, these models are often considered to operate in a class-transductive setting [157]. This different underlying hypothesis is more restrictive regarding the possible applications of the model. For this reason, we choose to

separate generative models from non generative models in results reported in tables 2.1 and 2.3. For f-GAN, we report the results with the f-CLSWGAN version with softmax as they are the best reported results in [162]. We do not report results for *instance*-transductive models as they rely on a quite different set of assumptions.

The model described in Section 2.4 is implemented[5] in PyTorch [113]. We report results for both variants $\boldsymbol{\theta} + \mathbf{I}$ (equations (2.22) and (2.23)) and $\boldsymbol{\theta} + \boldsymbol{\phi}$ (equations (2.24) and (2.25)), as well as results for the "full" model where the choice of the variant is considered to be a hyper-parameter. We train our models for 50 epochs using the ADAM optimizer [65], with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 0.001. All reported results for our models are the mean result over 10 runs of the model, with a different random initialization of the parameters each time. The importance of this step will be highlighted later in Section 2.7.1.

2.5.1 Zero-shot learning results

Table 2.1 reports results in a standard ZSL setting, where test samples belong to unseen classes and candidate classes consist of unseen classes only so that the reported accuracy is $A_{\mathcal{U}\to\mathcal{U}}$.

Both variants $\boldsymbol{\theta} + \mathbf{I}$ and $\boldsymbol{\theta} + \boldsymbol{\phi}$ outperform all non-generative models on two out of three datasets, and the best variant $\boldsymbol{\theta} + \mathbf{I}$ for AwA2 also outperforms all non-generative models. On all three datasets, the variant with the best validation score corresponds to the best final result, so model selection is quite robust. We therefore also include the result of the final model, where the choice of variant is considered to be a hyper-parameter. Interestingly, the final model also outperforms generative approaches on two datasets and is the highest performing model on average[6].

The fact that our model is not performing as well on AwA2 as on CUB and SUN compared to the other models is not really surprising: since this dataset is not a fine-grained dataset with a large number of classes, some of the proposed components such as the flexible semantic margin are not as relevant as on CUB or SUN.

Method	**CUB**	**SUN**	**AwA2**	Average[6]
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	41.8	46.7	49.7	46.1
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	53.5	61.5	68.9	61.3
ESZSL* [130]	53.9	54.5	58.6	55.6
SAE* [70]	33.3	40.3	54.1	42.6
DeViSE* [40]	52.0	56.5	59.7	56.0
SJE* [4]	53.9	53.9	61.9	56.6
ALE* [3]	54.9	58.1	62.5	58.5
$\text{SynC}_{\text{o-vs-o}}$* [22]	55.6	56.3	46.6	52.8
PSR [5]	56.0	61.4	63.8	60.4
Ours, $\boldsymbol{\theta} + \mathbf{I}$	61.4	62.2	67.9	63.8
Ours, $\boldsymbol{\theta} + \boldsymbol{\phi}$	**63.8**	**63.5**	61.5	62.9
Ours	**63.8**	**63.5**	67.9	**65.1**
Generative models†				
GFZSL*† [153]	49.3	60.6	63.8	57.9
SE-GZSL† [152]	59.6	63.4	69.2	64.0
f-GAN† [162]	57.3	60.8	68.2	62.1
GMMM-ZSL† [18]	59.4	60.1	**69.9**	63.1

Table 2.1 – Per class accuracy $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ measured for different ZSL models on 3 datasets. Results reported in [161] are marked with * next to the model's name. Other results are reported from their respective cited articles, except for $\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$ and $\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$ which were independently implemented. The generative models, marked with †, rely on stronger hypotheses as explained in Section 1.2.4. Our results are averaged over 10 runs.

Variant	Flexible margin	(Mahalanobis distance)	Partial normalization	Relevance weighting	**Score**
$\boldsymbol{\theta} + \boldsymbol{\phi}$		✓	✓	✓	**63.8**
	✓	-	✓	✓	61.7
		-	✓	✓	61.8
		✓	-	✓	57.6
		✓	✓	-	61.7
		-	✓	-	61.0
		-	-	-	56.6
$\boldsymbol{\theta} + \mathbf{I}$		✓	✓	✓	**61.3**
		-	✓	✓	61.1
		✓	-	✓	55.3
		✓	✓	-	60.0
		-	✓	-	60.1
		-	-	-	55.0

Table 2.2 – Ablation study on the CUB dataset for the two variants of our model $\boldsymbol{\theta} + \boldsymbol{\phi}$ and $\boldsymbol{\theta} + \mathbf{I}$. Results are averaged over 10 runs.

2.5.2 Ablation study

We perform an ablation study in order to evaluate the impact of the individual components of our proposed approach. We successively deactivate the flexible semantic margin by setting $\sigma_{\mathbf{M}}$ to 0 in Equation (2.7) so that $m(c, c') = \mu_{\mathbf{M}}$; the partial normalization by setting ρ to 0 in Equation (2.11) so that $\mathbf{\Psi}_\rho(\mathbf{v}) = \mathbf{v}$; and the relevance weighting scheme by setting all the weights v_n to 1 in Equation (2.20) so that all samples have the same weight. We also evaluate the impact of taking attribute correlations into account to estimate class similarities by setting $\mathbf{\Sigma}^{-1}$ to the identity matrix $\mathbf{I}_K$ in the Mahalanobis distance in Equation (2.4), so that semantic distances correspond to euclidean distances. Active hyper-parameters are re-evaluated on the validation set. Each reported result is the average result over 10 runs with different random initializations to avoid reporting artifacts from random noise.

Table 2.2 shows the results on the CUB dataset for both variants $\boldsymbol{\theta} + \boldsymbol{\phi}$ and $\boldsymbol{\theta} + \mathbf{I}$. Partial normalization has the largest impact in both cases: for the $\boldsymbol{\theta}+\boldsymbol{\phi}$, deactivating the partial normalization only makes the score drop from 63.8 to 57.6. Conversely, activating the partial normalization only increases the score from 56.6 to 61.0. Similar results can be observed with the $\boldsymbol{\theta} + \mathbf{I}$ variant. The flexible semantic margin and relevance weighting also significantly increase the final score. The three components work well together, as their combined impact is better than the sum of their marginal impacts, particularly in the $\boldsymbol{\theta} + \boldsymbol{\phi}$ variant.

As explained in Section 2.4, deactivating all three components and using the $\boldsymbol{\theta} + \mathbf{I}$ variant yields a model very close to DeViSE. The bottom line in Table 2.2 corresponds to this setting and has comparable results. The slight increase in performance compared to the result reported in [161] (55.0 compared to 52.0 in [161] as reported in Table 2.1) may be attributed to the fact that the fixed-value $\mu_{\mathbf{M}}$ of the margin is still considered to be a hyper-parameter in our approach, and may provide additional regularization. Our reproduction of DeViSE in Table 2.5 of Section 2.7.1 in which the size of the margin is not a hyperparameter produces a closer score of 52.6.

Method	CUB			SUN			AwA2			$\bar{\mathcal{H}}$[6]
	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	
Non generative approaches										
Ridge$_{\mathcal{V}\to\mathcal{S}}$	11.0	52.3	18.2	12.4	23.5	16.3	4.4	86.8	8.3	14.3
Ridge$_{\mathcal{S}\to\mathcal{V}}$	23.7	52.8	32.7	19.6	32.5	24.4	30.3	82.0	**44.3**	**33.7**
ESZSL* [130]	12.6	63.8	21.0	11.0	27.9	15.8	5.9	77.8	11.0	15.9
SAE* [70]	7.8	54.0	13.6	8.8	18.0	11.8	1.1	82.2	2.2	9.2
DeViSE* [40]	23.8	53.0	32.8	16.9	27.4	20.9	17.1	74.7	27.8	27.2
SJE* [4]	23.5	59.2	33.6	14.7	30.5	19.8	8.0	73.9	14.4	22.6
ALE* [3]	23.7	62.8	34.4	21.8	33.1	26.3	14.0	81.8	23.9	28.2
SynC* [22]	11.5	70.9	19.8	7.9	43.3	13.4	10.0	90.5	18.0	17.0
PSR [5]	24.6	54.3	33.9	20.8	37.2	**26.7**	20.7	73.8	32.3	31.0
Ours, $\boldsymbol{\theta + \mathbf{I}}$	30.4	64.0	**41.2**	21.3	34.1	26.2	17.6	79.8	28.9	32.1
Ours, $\boldsymbol{\theta + \phi}$	26.0	65.8	37.3	22.0	33.9	**26.7**	14.8	78.0	24.9	29.6
Ours	30.4	64.0	**41.2**	22.0	33.9	**26.7**	17.6	79.8	28.9	32.3
Generative approaches†										
GFZSL*† [153]	0.0	45.7	0.0	0.0	39.6	0.0	2.5	80.1	4.8	1.6
SE-GZSL† [152]	41.5	53.3	46.7	40.9	30.5	34.9	58.3	68.1	**62.8**	48.1
f-GAN† [162]	43.7	57.7	49.7	42.6	36.6	**39.4**	57.9	61.4	59.6	**49.6**
GMMM-ZSL† [18]	49.1	55.9	**52.3**	39.7	37.7	38.7	46.3	77.3	57.3	49.4

Table 2.3 – GZSL results for different ZSL models on 3 datasets. Results reported in [161] are marked with * next to the model's name. Other results are reported from their respective cited articles, except for Ridge$_{\mathcal{V}\to\mathcal{S}}$ and Ridge$_{\mathcal{V}\to\mathcal{S}}$ which were independently implemented. The generative models, marked with †, rely on stronger hypotheses as explained in Section 1.2.4. Our results are averaged over 10 runs.

2.5.3 Generalized zero-shot learning results

Table 2.3 reports results in a GZSL setting, where test samples (and therefore candidate classes) can belong to either seen or unseen classes. We measure $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ as well as their harmonic mean $\mathcal{H}$ (Section 1.4).

As observed in [163] and [24], and mentioned in Section 1.4, for non generative approaches there is usually a strong imbalance in favor of the seen classes, as $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ is much higher than $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$, which penalizes the final score $\mathcal{H}$. As a result, the best GZSL models are typically those with the best $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$, which happens to be the case for our model on two out of three datasets. We will implement a simple solution to reduce this seen-unseen class imbalance and thus improve the GZSL score for all models

[5] Our demo code is available at `https://github.com/yannick-lc/iccv2019-triplet-loss`.

[6] Reporting the score averaged over several datasets is debatable, but this enables to highlight the fact that the model outperforms other methods even if it does not obtain the best score on all datasets.

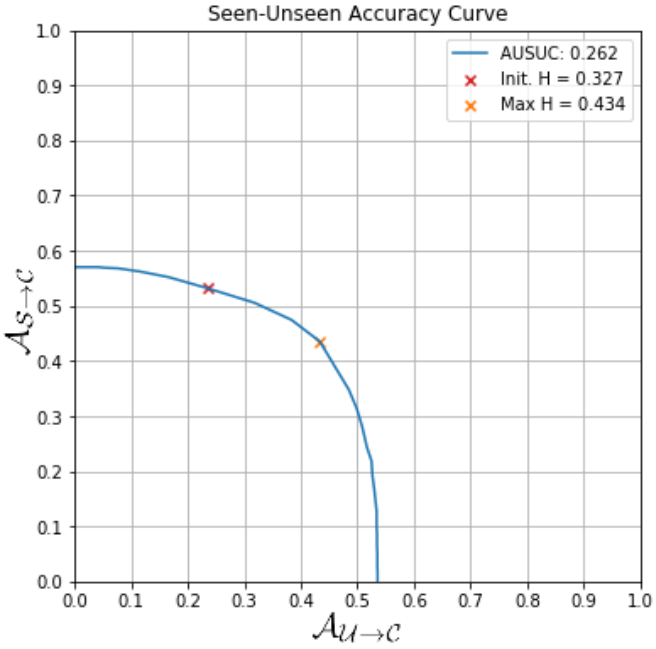

Figure 2.7 – Seen-Unseen Accuracy Curve for the Ridge$_{\mathcal{S}\to\mathcal{V}}$ model evaluated on the CUB dataset. When γ=0, we obtain an $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ of 23.7 and an $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ of 52.8, resulting in an $\mathcal{H}$ of 32.7 as in Table 2.3. When $\gamma = +\infty$, only unseen classes can be predicted and $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ is maximal and equal to 53.5, which corresponds to the ZSL score $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ from Table 2.1. When $\gamma = -\infty$, only seen classes can be predicted and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ is maximal. The best possible trade-off between the two occurs when both $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ are approximately equal to 43.4, resulting in a maximum theoretical $\mathcal{H}$ of 43.4. The AUSUC is the area under the curve.

in Section 2.6.

On the other hand, generative models typically suffer much less from performance imbalance between seen and unseen classes, as generated samples from unseen classes are most often used to train standard supervised classifiers alongside samples from seen classes. One notable exception is GFZSL, which does not train a standard classifier but instead uses the estimated distributions of unseen classes in the visual space to take a maximum likelihood approach for prediction (Section 1.2.4). This may explain the poor performance of GFZSL in a GZSL setting.

2.6 Addressing the seen-unseen classes gap

2.6.1 Calibration

We now propose a simple process to address the gap in performance between seen and unseen classes in a GZSL setting evidenced in Table 2.3. This process is based on the idea that a small degradation on $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ could result in a large improvement on $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$.

As mentioned in Section 1.4, [24] proposed a simple mechanism, called calibrated stacking, in which a weight γ is introduced to constrain the model to predict unseen classes more often when $\gamma > 0$, or seen classes more often when $\gamma < 0$:

$$\hat{y} = \underset{c\in\mathcal{C}}{\operatorname{argmax}}\ f(\mathbf{x}, \mathbf{s}_c) - \gamma \mathbb{1}[c \in \mathcal{C}^{\mathcal{S}}] \tag{2.27}$$

However, in [24], no specific value is attributed to γ. Instead, all possible values[7] of γ from $-\infty$ to $+\infty$ are used to estimate the impact on $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$. When $\gamma = -\infty$ (or rather, when γ is negative and its absolute value very large), the model can only predict seen classes; as a result $\mathcal{A}_{\mathcal{S}}$ is maximal and $\mathcal{A}_{\mathcal{U}}$ is equal to 0, as illustrated in Figure 2.7. Similarly, when $\gamma = +\infty$, the model can only predict unseen classes: $\mathcal{A}_{\mathcal{U}}$ is maximal and $\mathcal{A}_{\mathcal{S}}$ is 0. All values in between correspond to different trade-offs between $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$, with $\gamma = 0$ corresponding to the "default" trade-off. All the values of γ are used to draw a parametric plot with the value of $\mathcal{A}_{\mathcal{U}}$ on the x-axis and the value of $\mathcal{A}_{\mathcal{S}}$ on the y-axis (Figure 2.7). The area under this curve, the Area Under Seen-Unseen Accuracy Curve or AUSUC, can be used to measure the performance of the model similarly to the area under a Receiver Operating Characteristic curve [38].

It is important to emphasize that this process is applied *a posteriori* to an already trained ZSL model in order to evaluate its performance. In particular, the impact of γ is measured on the *test set*. As a result, this process is not directly applicable to balance the performance of the model between seen and unseen classes.

By contrast, we propose to employ a similar process, i.e. a weight γ used to adjust the estimated compatibility depending on the nature of each prototype $\mathbf{s}_c$, in order to balance $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ *without*

[7] In practice, it is not necessary (nor feasible) to test all possible values. It is sufficient to have a largest value large enough to classify all test samples as unseen classes, and a smallest value small enough to classify all test samples as seen classes.

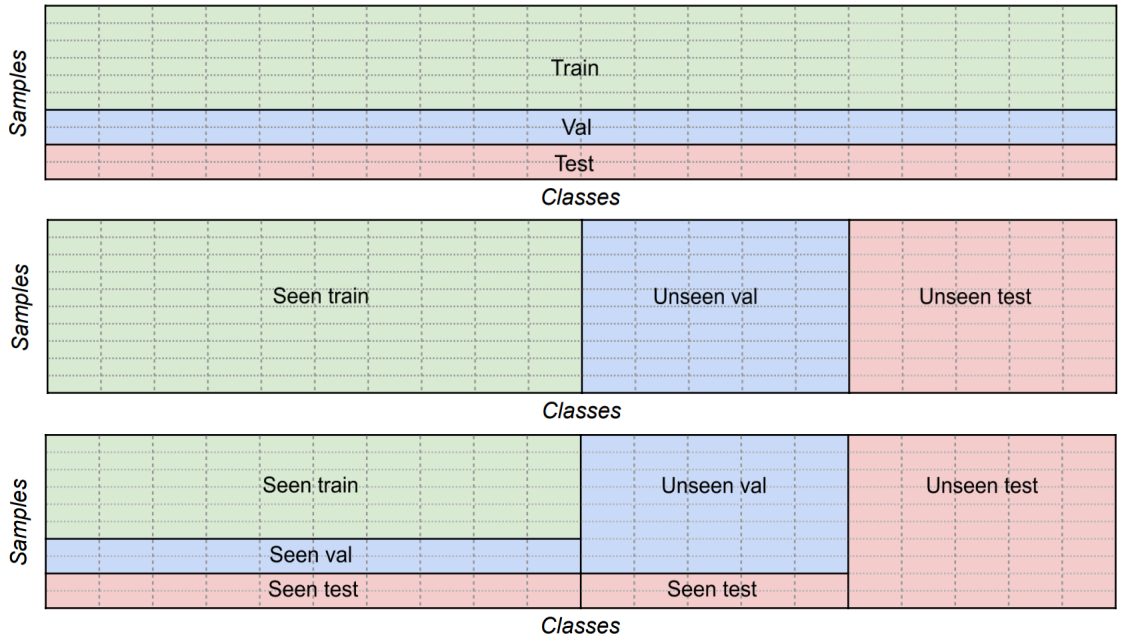

Figure 2.8 – Training-validation-testing splits in different settings. Each column represents a class, and each small rectangle a sample of this class (classes are represented as balanced in this figure, even though this is not necessarily the case). *Top*: standard ML split, with respect to samples. *Middle*: "classical" ZSL split, with respect to classes. *Bottom*: proposed GZSL split.

having access to the test set. We call this process the *calibration* process. This requires to select a good value for γ using the training set only. This is achieved with a cross-validation specific to the GZSL setting, as we are about to describe.

In a standard supervised machine learning setting, the training set is typically further divided into a "true" training set, used to learn the parameters $\mathbf{w}$ of the model, and a validation set, employed to select the best hyper-parameter(s) λ of the model – in some cases, k-fold cross-validation is used so there is not a single validation set, but this distinction is not important here and the proposed approach can easily be extended to this setting. In a multi-class classification setting with enough samples per class, there are usually samples from all classes in all splits. This corresponds to the top part in Figure 2.8.

On the other hand, in a ZSL setting, the training / validation / testing split is done with respect to the classes as opposed to the samples: a set of classes is used for training, a disjoint set for validation, and a final set disjoint from the first two is employed for testing. This corresponds to the middle part in Figure 2.8.

In GZSL, a fraction (typically 20% [163]) of the samples from the seen classes (classes in the training or validations sets) is not used for training or validation and is kept for the testing phase in order to evaluate $\mathcal{A}_{\mathcal{S}}$. We refer to this set as the *seen test set*. We emphasize that here *seen* simply means that the samples belong to seen classes, and *not* that these specific instances have been used during training or previously seen by the model.

In order to be able to cross-validate hyper-parameters in a GZSL setting, we keep an additional 20% of the remaining training set from being used in training, and we use these samples as samples from seen classes during validation. We refer to this set as the *seen validation set*. This way, it is possible to evaluate the impact of hyper-parameters on both $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ on the validation set(s), without using any sample from the seen and unseen test sets. All the different sets and splits are illustrated in Figure 2.8.

It is now possible to determine the optimal value of γ in Equation (2.27) with the following process.

(1) The dataset is split as explained previously and as illustrated by the bottom part of Figure 2.8. A standard ZSL model is trained using the samples in the seen train set. (2) Its ZSL accuracy $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ can be evaluated on the unseen validation set, so that hyper-parameters can be selected accordingly. (3) The unseen validation set can also be employed to measure $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$; on the other hand, the seen validation set can be used to measure $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$. We can then test different values of γ and measure their impact on $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$. We emphasize that no retraining of the model is required, as γ only affects the prediction phase of a trained model (Equation (2.27)). The optimal value of γ can finally be selected so as to maximize the harmonic mean $\mathcal{H}$ of $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$, or any other relevant GZSL metric.

(4) The ZSL model is subsequently retrained using the seen train set, seen validation set and unseen validation set as the new training set. (5) The class compatibilities $f(\mathbf{x}_n, \mathbf{s}_c)$ are evaluated for all testing samples $\mathbf{x}_n$, in the unseen test set as well as the seen test set, with respect to all class prototypes $\mathbf{s}_c$. (6) For class prototypes $\mathbf{s}_c$ in seen classes $c \in \mathcal{C}^{\mathcal{S}}$, the constant value γ selected previously is subtracted from the corresponding compatibilities in accordance with Equation (2.27) to predict $\hat{y}$. (7) $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ can finally be measured on respectively the unseen test set and the seen test set to obtain the final score $\mathcal{H}$.

The effectiveness of this selected γ to reduce the imbalance between $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ will be evaluated in Section 2.7.

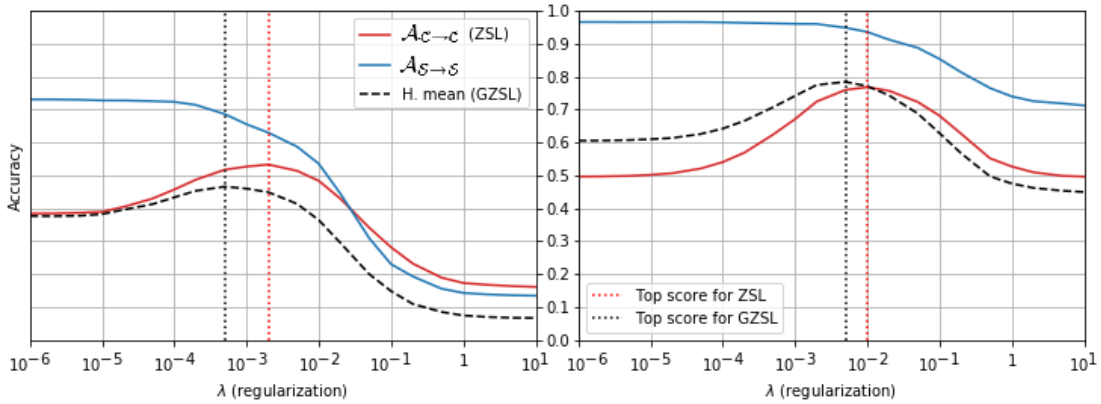

Figure 2.9 – Illustration of how the regularization parameter λ of the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model affects the accuracies on samples from seen and unseen classes $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ (blue) and $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ (red), as measured on the test sets of CUB *(left)* and AwA2 *(right)*. The optimal value for λ is not the same in a ZSL setting, where performance is measured with $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ (red vertical dotted line), and in a GZSL setting, where performance is measured by the harmonic mean $\mathcal{H}$ of $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ (black vertical dotted line).

2.6.2 Hyper-parameter selection

In the usual direct stacking approach to GZSL (see Section 1.4), the hyper-parameters of the model are typically selected with a train / validation split similar to the middle one in Figure 2.8, before evaluating the model in a GZSL setting [24, 163]. We argue that hyper-parameters selected with this process may be suitable for the classical ZSL task but are not necessarily adapted to the GZSL task. We provide empirical evidence and theoretical insights to justify this view.

Figure 2.9 shows $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ *(red)* and $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ *(blue)* as a function of λ, the hyperparameter for a regularized linear model $\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$ as described in equations (1.24) and (1.33), measured on a ZSL validation split for the datasets CUB *(left)* and AwA2 *(right)*. In this setting, a larger value of λ means a stronger regularization. In both datasets, there is a value of λ that maximizes the ZSL score $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$, indicated by the red dotted vertical line. We call this value λ^*_{ZSL}. λ^*_{ZSL} seems to correspond to a local (or even a global) maximum. On the other hand, the overall tendency for $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ is to decrease as λ increases. This tendency is not a concern for the ZSL task, since we only consider samples from unseen classes. However, for the GZSL task, we want the best trade-off between $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ as measured by their harmonic mean $\mathcal{H}$.

Since the (red) curve for $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ seems much flatter around λ^*_{ZSL} than the (blue) curve for $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$, we can expect that a small decrease in λ will increase $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ more than it will decrease $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$. Since

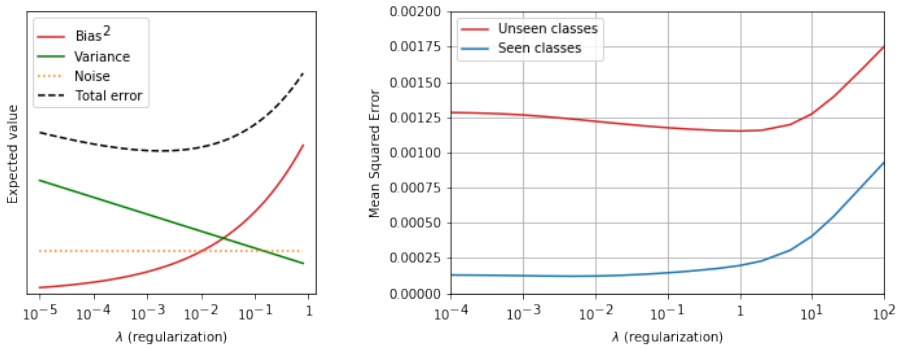

Figure 2.10 – *Left:* Illustration of the bias-variance decomposition. *Right:* Mean squared error of predicted attributes (averaged over attributes and samples) as a function of the regularization parameter λ with the Ridge$_{\mathcal{S}\to\mathcal{V}}$ model on the validation set of the AwA2 dataset.

$\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ are upper bounds of $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ (with equality only if we are able to perfectly distinguish samples from seen and unseen classes), we can expect a similar result on $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$, and thus an increase of their harmonic mean $\mathcal{H}$. The actual gains on the GZSL task obtained by decreasing λ compared to λ^*_{ZSL} will be quantified in Section 2.7.

The reason why λ affects $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{S}}$ in this way can be at least partly explained with the bias-variance decomposition. For a regression task, we generally assume that we are given a dataset $\mathcal{D} = (\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \{t_1, \ldots, t_N\})$, consisting of samples $(\mathbf{x}_n, t_n)$ independently drawn from a joint distribution $p(\mathbf{x}, t)$, such that $p(t|\mathbf{x}) = \mathcal{N}(t|h(\mathbf{x}), \sigma^2)$, where h is the true dependence function. For a prediction function $\hat{h}_\mathcal{D}$ estimated from $\mathcal{D}$ we can then decompose the expected squared error $\mathbb{E}_{\mathcal{D},\mathbf{x},t}[(t - \hat{h}_\mathcal{D}(\mathbf{x}))^2]$ on a new pair $(\mathbf{x}, t)$ drawn from the same distribution as:

$$\mathbb{E}_{\mathcal{D},\mathbf{x},t}[(t - \hat{h}_\mathcal{D}(\mathbf{x}))^2] = \sigma^2 + \Big(\mathbb{E}_{\mathcal{D},\mathbf{x}}[h(\mathbf{x}) - \hat{h}_\mathcal{D}(\mathbf{x})]\Big)^2 + \mathrm{var}_{\mathcal{D},\mathbf{x}}[\hat{h}_\mathcal{D}(\mathbf{x})] \tag{2.28}$$

where the first term is the intrinsic noise of the distribution, the second is the (squared) bias of the predictor $\hat{h}_\mathcal{D}$ and the third is the variance in the estimation of the predictor [12]. It can be shown [151] that for ridge regression the bias increases and the variance decreases with the regularization (hyper)parameter λ, as illustrated in Figure 2.10 (left). There is usually a value of λ that represents the best trade-off between bias and variance as measured by the expected total error.

In the case of ZSL, $\mathbf{x}$ corresponds to a visual instance and t to attribute(s) to be estimated from $\mathbf{x}$. The variance in the dataset $\mathcal{D}$ comes from both the differences between samples from the same

Dataset	Intra-class variance	Inter-class variance
CUB	138.0	231.9
AwA2	226.4	379.0
SUN	239.9	397.3

Table 2.4 – Intra-class and inter-class variance for several datasets. Intra-class variance is the mean squared distance between visual samples of a class and the mean of samples from this class, averaged over all classes. Inter-class variance is the mean squared distance between all samples and the mean sample.

class (intra-class variance) and from the differences between classes (inter-class variance). Intra-class variance is usually significantly smaller than inter-class variance in ZSL datasets (Table 2.4). Therefore, most of the variance in Equation (2.28) can be attributed to the choice of the training classes $\mathcal{C}^{\mathcal{S}}$.

For samples from unseen classes, the bias-variance decomposition applies and there exists a λ corresponding to the best trade-off between the two. This is evidenced in Figure 2.10 (right), where the red curve shows the mean squared error (MSE) in the predictions of attributes from unseen classes as a function of λ, for a regularized linear model on a validation split of AwA2. This curve is similar to the dotted black curve on the right, corresponding to the total error (bias plus variance plus noise).

On the other hand, for samples from seen classes, the variance in the dataset $\mathcal{D}$ – and thus in the predictor $\hat{h}_{\mathcal{D}}$ – attributable to the choice of the training classes is much smaller since, by definition, these seen classes must be present in the training dataset. This allows to better estimate attributes from seen classes and most of the expected error therefore comes from the intrinsic noise and the bias. Thus, the expected error mostly increases with λ, as evidenced by the blue curve in Figure 2.10 (right). This curve is similar to the sum of the red and orange curves on the right, corresponding to the variance plus noise.

If we reasonably assume that the accuracy of predictions for samples from a given class depends on how well we estimate their attributes, this explains both why predictions are better for samples from seen classes than from unseen classes and why their behavior with respect to λ is different. Although we assumed that λ correspond to the regularization hyper-parameter from a ridge regression model, this reasoning is applicable to any model where a (set of) hyper-parameter(s) enables to increase the bias and reduce the variance.

We then suggest the following procedure to select the optimal value of λ or any other set of hyper-parameters in a GZSL setting: we repeat the protocol described in Section 2.6.1 to select the calibration weight γ and we take the value of λ which gives the best result for the harmonic mean $\mathcal{H}$ of $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ on the validation set *after* having subtracted γ from the compatibilities of seen classes as in Equation (2.27). The rest of the process is identical: we retrain the ZSL model on the seen train, seen validation and unseen validation sets (Figure 2.8, bottom) with the hyperparameter λ that we just selected, we compute compatibilities for the test set, subtract γ from the compatibilities of seen classes and compute the resulting GZSL score.

2.7 Experimental evaluation of the calibration process

We now evaluate the impact of the proposed calibration and hyper-parameters selection process.

2.7.1 Reproduction of results

In addition to our proposed model from Section 2.4, we would like to evaluate the effectiveness of this process on other methods from tables 2.1 and 2.3. With this goal in mind, we independently re-implement ESZSL [130], SAE [70], DeViSE [40], SJE [4], ALE [2] and SynC [22] from their descriptions in their respective original publications. We provide the following details and clarifications: for SAE, during the prediction phase, distances are evaluated in the visual space as it yields better results. For ALE, DeViSE and SJE, we add a hyperparameter λ weighing a regularization term of the form $\|\mathbf{W}\|_2^2$, $\mathbf{W}$ being the parameters of the models used in their bi-linear compatibility functions $f_{\mathbf{W}}(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \mathbf{W}\mathbf{s}$. For SynC, we use the "structured loss" version described in [22] as it yields better results.

We mostly use the same experimental protocol as in [161] and Section 2.5: we perform experiments on CUB, SUN and AwA2, using the same "proposed" ZSL splits as [161]. Visual features are extracted from a pre-trained ResNet-101 model. We do not apply 10-crop for now to be as close as possible to the experimental setting in [161]. Result with 10-crop are provided later in Table 2.8. Semantic prototypes are $\ell 2$ normalized. For models based on iterative numerical optimization, namely DeViSE, SJE, ALE and SynC, we perform training and evaluation multiple times with different random initializations of the parameters. We train and evaluate these models 5 times unless stated otherwise.

We compare our reproduced results with results reported in [161] in tables 2.5 and 2.6 in the

Method	**Reported in [161]**	**Mean**	std	min	max
CUB dataset					
ESZSL	53.9	34.9	±0.0	34.9	34.9
SAE	33.3	53.3	±0.0	53.3	53.3
DeViSE	52.0	52.9	±1.0	51.4	54.1
SJE	53.9	49.4	±1.1	48.4	50.8
ALE	54.9	54.6	±0.7	53.3	55.4
SynC	55.6	58.8	±0.6	57.9	59.7
SUN dataset					
ESZSL	54.5	16.0	±0.0	16.0	16.0
SAE	40.3	61.0	±0.0	61.0	61.0
DeViSE	56.5	61.6	±0.3	61.3	61.9
SJE	53.7	55.3	±1.1	53.9	56.5
ALE	58.1	59.4	±0.3	58.8	59.7
SynC	56.3	56.6	±1.8	54.1	59.1
AwA2 dataset					
ESZSL	58.6	50.8	±0.0	50.8	50.8
SAE	54.1	62.8	±0.0	62.8	62.8
DeViSE	59.7	62.1	±1.6	58.9	63.2
SJE	61.9	62.3	±1.2	60.1	63.4
ALE	62.5	62.9	±2.3	59.1	65.6
SynC	46.6	58.1	±0.8	56.9	59.3

Table 2.5 – Reproduction of ZSL results from [163, 161], as measured by $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$. "Mean" is the mean result over 5 runs with different random initializations. "std", "min" and "max" are the respective corresponding standard deviation minimal score and maximal score obtained over these 5 runs.

Method	**Reported in [161]**			**Reproduced**		
	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$
CUB dataset						
ESZSL	12.6	63.8	21.0	10.5 ±0.0	61.8 ±0.0	17.9 ±0.0
SAE	7.8	54.0	13.6	16.7 ±0.0	56.3 ±0.0	25.7 ±0.0
DeViSE	23.8	53.0	32.8	24.8 ±0.7	58.1 ±1.3	34.8 ±0.5
SJE	23.5	59.2	33.6	19.7 ±0.7	53.2 ±1.5	28.7 ±0.8
ALE	23.7	62.8	34.4	25.3 ±0.8	59.3 ±1.3	35.4 ±0.7
SynC	11.5	70.9	19.8	22.6 ±1.0	62.3 ±1.3	33.2 ±1.0
SUN dataset						
ESZSL	11.0	27.9	15.8	4.3 ±0.0	15.3 ±0.0	6.7 ±0.0
SAE	8.8	18.0	11.8	20.6 ±0.0	31.9 ±0.0	25.1 ±0.0
DeViSE	16.9	27.4	20.9	21.5 ±0.4	30.4 ±0.2	25.2 ±0.2
SJE	14.7	30.5	19.8	19.6 ±0.7	36.4 ±0.5	25.5 ±0.6
ALE	21.8	33.1	26.3	23.0 ±0.3	32.3 ±0.4	26.9 ±0.3
SynC	7.9	43.3	13.4	16.2 ±0.8	28.4 ±1.0	20.6 ±0.8
AwA2 dataset						
ESZSL	5.9	77.8	11.0	26.6 ±0.0	79.8 ±0.0	39.9 ±0.0
SAE	1.1	82.2	2.2	17.6 ±0.0	90.2 ±0.0	29.5 ±0.0
DeViSE	17.1	74.7	27.8	9.7 ±2.5	89.8 ±0.8	17.4 ±4.1
SJE	8.0	73.9	14.4	17.4 ±1.7	85.8 ±0.5	28.9 ±2.3
ALE	14.0	81.8	23.9	23.0 ±0.3	32.3 ±0.4	26.9 ±0.3
SynC	10.0	90.5	18.0	18.7 ±0.3	83.4 ±0.4	30.6 ±0.3

Table 2.6 – Reproduction of GZSL results from [163, 161], as measured by $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$, $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ and $\mathcal{H}$. We report the mean result as well as the standard deviation over 5 runs with different random initializations

respective ZSL and GZSL settings. In Table 2.5, we report the mean per class accuracy over 5 runs of each model, as well as the associated standard deviation, and the minimum and maximum accuracy obtained during these runs. In Table 2.6, we report the mean $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$, $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ and $\mathcal{H}$ over 5 runs as well as the associated standard deviations.

Our first observation is that there can be variance in the measured accuracy depending on the random initialization of the parameters. The most striking example in a ZSL setting occurs with ALE on the AwA2 dataset, with an accuracy ranging from a mediocre score – compared to the other models – of 59.1% to the absolute best score of 65.6% over 5 runs. This highlights the importance of reporting a score averaged over several random initializations to make comparisons more reliable. We therefore recommend as a good practice to run each model 5 times or more depending on the computational resources and time available. Note that there is no variance in the results for ESZSL and SAE since these models are based on closed-form solutions and do not use stochastic optimization. This is also true for other models such as Ridge$_{\mathcal{V}\to\mathcal{S}}$or Ridge$_{\mathcal{S}\to\mathcal{V}}$.

Regarding the closeness of our reproduction, in a ZSL setting (Table 2.5), DeViSE, SJE, ALE and SynC tend to match results from [161], with a few exceptions. On the other hand, we obtain significantly different results for ESZSL and SAE. For ESZSL, our results are usually significantly below those from [161] in this experimental setting. However, we obtained results comparable to [161] if semantic prototypes are not $\ell 2$-normalized. For the sake of consistency, we nonetheless chose to always report results with normalized attributes. On the contrary, for SAE, our results are usually significantly above these from [161]. This is likely due to our choice to perform comparisons in the visual space as opposed to the semantic space: we found that results for SAE were close to Ridge$_{\mathcal{S}\to\mathcal{V}}$ in the former case, and close to the Ridge$_{\mathcal{V}\to\mathcal{S}}$ in the latter case, which seems to match results from [161].

Similar trends can be seen in a GZSL setting, as reported in Table 2.6.

2.7.2 Results of the proposed approach

We now evaluate the impact of the proposed calibration and hyper-parameter(s) selection process. Table 2.7 shows results on the models reproduced in Section 2.7.1 as well as on Ridge$_{\mathcal{S}\to\mathcal{V}}$ and Ridge$_{\mathcal{V}\to\mathcal{S}}$. *"No calibration"* indicates that the same protocol as in Section 2.7.1 is used: the models

Method	**No calibration**			**Calibration**			**Cal. + hyper-p.**		
	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$
CUB dataset									
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	11.0	52.3	18.2	38.2	31.2	34.3	39.3	32.4	35.5
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	23.7	52.8	32.7	45.8	38.6	41.9	45.4	41.7	43.5
ESZSL	10.5	61.8	17.9	32.6	34.8	33.7	34.2	33.7	33.9
SAE	16.7	56.3	25.7	41.6	44.7	43.1	41.6	44.7	43.1
DeViSE	24.8	58.1	34.8	47.2	41.1	44.0	43.1	43.7	43.4
SJE	19.7	53.2	28.7	41.5	41.1	41.3	45.2	44.2	44.7
ALE	25.3	25.9	35.4	48.9	43.3	46.0	48.7	45.0	46.8
SynC	22.6	62.3	33.2	48.7	44.0	46.2	48.8	46.5	47.6
Average	19.3	52.8	28.3	43.1	39.8	41.3	43.3	41.5	**42.3**
SUN dataset									
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	12.4	23.5	16.3	33.7	17.8	23.3	33.9	20.0	25.2
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	19.6	32.5	24.4	44.9	27.6	34.2	46.9	27.7	34.8
ESZSL	4.3	15.3	6.7	10.5	11.9	11.1	10.5	11.9	11.1
SAE	20.6	31.9	25.1	45.0	27.4	34.1	47.7	27.4	34.8
DeViSE	21.5	30.4	25.2	46.1	23.8	31.4	46.1	23.8	31.4
SJE	19.6	36.4	25.5	45.2	28.5	34.9	45.1	29.1	35.3
ALE	23.0	32.3	26.9	46.7	26.2	33.6	46.7	26.2	33.6
SynC	16.2	28.4	20.6	35.1	23.1	27.9	35.1	23.1	27.9
Average	17.1	28.8	21.3	38.4	23.3	28.8	39.0	23.6	**29.3**
AwA2 dataset									
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	4.4	86.8	8.3	36.7	66.6	47.3	37.3	67.4	48.7
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	30.3	82.0	44.3	52.0	79.1	62.7	53.2	80.3	64.0
ESZSL	26.6	79.8	39.9	50.4	57.2	53.6	47.1	62.5	53.7
SAE	17.6	90.2	29.5	48.0	80.7	60.2	48.0	80.7	60.2
DeViSE	9.7	89.8	17.4	40.6	83.0	54.6	40.6	83.0	54.6
SJE	17.4	85.8	28.9	46.4	78.2	58.2	46.2	81.8	59.0
ALE	15.9	87.4	26.9	42.8	80.1	55.8	42.8	80.1	55.8
SynC	18.7	83.4	30.6	49.4	79.7	61.0	49.4	79.7	61.0
Average	17.6	85.6	28.2	45.8	75.6	56.7	45.6	76.9	**57.1**

Table 2.7 – GZSL results without calibration, with calibration, and with calibration and hyper-parameters specific to the GZSL task. Result are averaged over 5 runs.

are trained in a ZSL setting, and then evaluated with the same hyper-parameters in a GZSL setting. *"Calibration"* indicates that we use the calibration process described in Section 2.6.1, while hyper-parameters are still selected in a ZSL setting. Finally, *"Cal. + hyper-p."* indicates that calibration is used, and hyper-parameters are selected using the process described in Section 2.6.2.

As expected, the calibration process is effective in reducing the gap between $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$: on the CUB dataset, the average values of $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ evaluated over the 8 models are respectively 19.3 and 52.8 without calibration, and 43.1 and 39.8 with calibration, which enables the average $\mathcal{H}$ to increase from 28.3 to 41.3. On AwA2, a similar effect can be observed. Even though the gap between $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ is still important, it is much less pronounced than without calibration which enables to double the average $\mathcal{H}$. Interestingly, on the SUN dataset, the reduction in the gap between $\mathcal{A}_{\mathcal{U}}$ and $\mathcal{A}_{\mathcal{S}}$ is not as pronounced: the models tend to over-predict unseen classes instead of seen classes before calibration, so the increase in $\mathcal{H}$ is less striking.

Choosing hyper-parameters specific to the GZSL task enables to further increase the final score on all datasets, although the marginal impact of this step is not as dramatic. As a side note, some models have the exact same scores in the *"Calibration"* and *"Cal. + hyper-p."* columns, for instance SAE when evaluated on the CUB dataset. This happens when the hyper-parameter(s) selected for the GZSL task are the same as the one selected for the ZSL task, leading to the same score[8].

Finally, in Table 2.8, we compare the GZSL results of these models to our proposed model(s) from Section 2.4, with calibration and hyper-parameters selected specifically for the GZSL task. For fair comparison, we use visual features extracted using 10-crop for all reproduced approaches. The proposed process enables to increase the $\mathcal{H}$ score of our proposed model by more than 18 points on average. Our model outperforms all non generative approaches on two out of three datasets, and outperforms them "on average" as well.

We also include results with generative approaches in Table 2.8, even though as stated earlier, these approaches are typically based on more restrictive hypotheses. We do not apply calibration to the generative approaches as they usually do not suffer from the performance gap between seen and unseen classes. Interestingly, our proposed model also outperforms generative approaches on the CUB

[8]One may still expect some minor differences caused by random noise. However, for the sake of reproducibility, we use a fixed set of random seeds, which leads to the same results.

dataset, as well as "on average". As a side note, it is also interesting to notice that using visual features extracted with 10-crop enables an increase of 1.6 points on average on the evaluated GZSL models.

2.8 Discussion

We proposed a ZSL model based on the triplet loss aiming to address some of the usual limitations of such approaches. We also proposed a simple process to fairly evaluate standard ZSL models in a GZSL setting, by reducing the performance gap between $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$. Under these conditions, our proposed model outperforms all non generative approaches, and gets performance close to or even better than recent generative approaches, even though the latter may rely on more restrictive hypotheses.

Our proposal relies on a number of design choices. There could be reasonable alternative possibilities for many of these components. For example, the architecture could be modified to include non-linearities; higher order statistics could be used to measure the distances and thus the "representativeness" in the visual space, similarly to the second order statistics employed to measure distances between classes in the semantic space; inter-class distances could be computed using both the semantic and the visual space, etc.

Experiments have been conducted for some of these alternative possibilities. Most of the time, we chose what appeared to be the simplest design choice if no significant difference in performance was observed. An example of such a more complicated approach consists in using a calibration hyper-parameter γ_c *per class* c instead of a unique γ as in Equation (2.27). However, since it did not lead to a measurable improvement in the GZSL score, we chose the simpler approach described in Section 2.6.1. Nonetheless, many of these alternate design choices are yet to be explored, as the possible combinations of such choices are too numerous to be exhaustively tested.

Method	CUB			SUN			AwA2			$\bar{\mathcal{H}}^6$
	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	$\mathcal{A}_\mathcal{U}$	$\mathcal{A}_\mathcal{S}$	$\mathcal{H}$	
Non generative approaches, without calibration										
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	11.0	52.3	18.2	12.4	23.5	16.3	4.4	86.8	8.3	14.3
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	23.7	52.8	32.7	19.6	32.5	24.4	30.3	82.0	**44.3**	**33.7**
ESZSL* [130]	12.6	63.8	21.0	11.0	27.9	15.8	5.9	77.8	11.0	15.9
SAE* [70]	7.8	54.0	13.6	8.8	18.0	11.8	1.1	82.2	2.2	9.2
DeViSE* [40]	23.8	53.0	32.8	16.9	27.4	20.9	17.1	74.7	27.8	27.2
SJE* [4]	23.5	59.2	33.6	14.7	30.5	19.8	8.0	73.9	14.4	22.6
ALE* [3]	23.7	62.8	34.4	21.8	33.1	26.3	14.0	81.8	23.9	28.2
SynC* [22]	11.5	70.9	19.8	7.9	43.3	13.4	10.0	90.5	18.0	17.0
PSR [5]	24.6	54.3	33.9	20.8	37.2	**26.7**	20.7	73.8	32.3	31.0
Ours, $\boldsymbol{\theta} + \mathbf{I}$	30.4	64.0	**41.2**	21.3	34.1	26.2	17.6	79.8	28.9	32.1
Ours, $\boldsymbol{\theta} + \boldsymbol{\phi}$	26.0	65.8	37.3	22.0	33.9	**26.7**	14.8	78.0	24.9	29.6
Ours	30.4	64.0	**41.2**	22.0	33.9	**26.7**	17.6	79.8	28.9	32.3
Non generative approaches, with calibration										
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	41.7	38.2	39.8	35.2	21.5	26.7	38.2	68.3	49.0	38.5
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	48.5	46.8	47.7	45.9	30.0	36.3	54.8	80.0	65.1	49.7
ESZSL** [130]	42.4	35.6	38.7	11.0	12.7	11.8	52.1	56.9	54.4	35.0
SAE** [70]	44.7	48.0	46.3	43.1	30.8	35.9	49.9	82.8	62.3	48.2
DeViSE** [40]	46.9	38.7	42.4	48.8	24.3	32.5	40.7	85.0	55.0	43.3
SJE** [4]	48.6	45.0	46.7	47.7	29.9	**36.8**	46.1	83.3	59.4	47.6
ALE** [3]	52.3	46.7	49.4	48.9	27.1	34.9	43.7	81.7	56.9	47.1
SynC** [22]	49.5	48.3	48.9	35.1	23.1	27.9	50.8	81.7	**62.6**	46.5
Ours, $\boldsymbol{\theta} + \mathbf{I}$	55.8	48.1	51.6	47.9	28.1	35.4	48.5	83.2	61.3	49.4
Ours, $\boldsymbol{\theta} + \boldsymbol{\phi}$	53.8	52.3	**53.0**	46.5	30.4	**36.8**	45.4	77.9	57.3	49.0
Ours	53.8	52.3	**53.0**	46.5	30.4	**36.8**	48.5	83.2	61.3	**50.4**
Generative approaches†										
GFZSL*† [153]	0.0	45.7	0.0	0.0	39.6	0.0	2.5	80.1	4.8	1.6
SE-GZSL† [152]	41.5	53.3	46.7	40.9	30.5	34.9	58.3	68.1	**62.8**	48.1
f-GAN† [162]	43.7	57.7	49.7	42.6	36.6	**39.4**	57.9	61.4	59.6	**49.6**
GMMM-ZSL† [18]	49.1	55.9	**52.3**	39.7	37.7	38.7	46.3	77.3	57.3	49.4

Table 2.8 – GZSL results without calibration, and with calibration and hyper-parameters specific to the GZSL task. Results from [161] are marked with * next to the model's name. Results with calibration were all obtained from our independent implementation, use 10crop visual features, and are averaged over 5 runs. The generative models, marked with †, rely on stronger hypotheses as explained in Section 1.2.4. Results for our model are averaged over 10 runs.

Chapter 3

Semantic representation for large scale zero-shot learning

Content

In Chapter 2, we considered ZSL datasets whose semantic prototypes consisted of binary or continuous attributes. However, when the number of classes becomes very large, it can be impractical to design high quality semantic representations using attributes. For instance, the CUB dataset comes with manually-defined 312-dimensional vectors of attributes, and performance of ZSL models drops

very quickly as the number of attributes decreases, as will be measured in Section 3.2.4. It may thus not be practicable to manually provide hundreds of such attributes for each class of a large scale dataset consisting of thousands of classes, such as the ImageNet dataset. As a result, in a large scale setting, it is common to use semantic prototypes which can be automatically obtained in an *unsupervised* way. These prototypes typically consist in word embeddings of class names, which use an embedding model pre-trained on a large text corpus. However, performance with this type of semantic representation is generally much lower than with manually-designed attributes, as will also be measured in Section 3.2.4.

In this chapter, we focus primarily on the impact of semantic representations in a large scale zero-shot learning setting. We propose several approaches for designing and making use of suitable semantic prototypes, with the aim of enabling a better trade-off between the high effort - high performance nature of manually designed attributes and the low effort - low performance nature of "unsupervised" semantic prototypes than the current *status quo*. The chapter is organized as follows: in Section 3.1, we argue that generic text corpora may not be optimal to produce embeddings suitable for zero-shot recognition. We thus propose to collect new datasets with more visually oriented textual content. We further propose to adapt existing word embedding models via some pre-processing steps in order to adequately leverage these corpora[1]. Performance of this method is evaluated in Section 3.2. In addition, this section provides a comparison of unsupervised embeddings with manual attributes, as well as a detailed error analysis. In Section 3.3, we argue that short descriptions may provide relevant information not contained in word embeddings for a reasonable annotation cost. We propose several ways to employ short sentences as semantic information for zero-shot learning. Section 3.4.1 provides an experimental evaluation of these approaches. Section 3.5 explores the combination of embeddings from class names and from short sentence descriptions as class prototypes. Finally, Section 3.6 provides a discussion on some design choices, limits of the proposed methods and currently unsuccessful tracks. Similarly to Chapter 2, we adopt the notations defined in Section 1.1.2 and summarized in Table 1.2.

[1]This work is the result of a collaboration with Adrian Popescu, Université Paris-Saclay, CEA, List.

3.1 Unsupervised semantic prototypes

As detailed in Section 1.3.2, in a large scale ZSL setting, it is customary to use word embeddings of class names to obtain class prototypes with close to zero annotation effort. This is made possible by the fact that word embedding models are trained in an *unsupervised* way, and thus do not require human-defined labels. For instance, in the Word2vec approach [102], the skip-gram objective aims to predict the words present in the context of a given word. If we consider a corpus of L sentences, where the l^{th} sentence consists of T_l words $\{w_1, \dots, w_{T_l}\}$, this can be achieved by minimizing

$$-\sum_{l=1}^{L}\sum_{t=1}^{T_l}\sum_{\substack{-S\leq i\leq S\\ i\neq 0}} \log\, p(w_{t+i}|w_t) \tag{3.1}$$

Here S is the size of the context window and determines how close to each other two words w_i, w_j need to be to be considered part of each other's context. $p(w_i|w_t)$ in Equation (3.1) is typically estimated using a 1-hidden layer fully connected neural network. Each unique word w is thus associated with an "input" vector $\mathbf{v}_w$ and an "output" vector $\mathbf{v}'_w$, and $p(w_i|w_t)$ is computed such that

$$p(w_i|w_t) = \frac{\exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_{w_i})}{\sum_w \exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_w)} \tag{3.2}$$

The input vector representation $\mathbf{v}_w$ can then be employed as the embedding of word w, and a semantic representation $\mathbf{s}_c$ for class c can be obtained by using the word embedding corresponding to its name. For classes whose name consists of several words, we can simply employ the average of the corresponding word embeddings as the semantic prototype – and repeat this process recursively for classes consisting of several multi-word lemmas as in the ImageNet dataset [33] (Appendix A.1).

The word embedding models are typically trained on large generic text corpora such as Wikipedia, Google News or CommonCrawl. Although these embeddings enable to effortlessly obtain semantic prototypes and thus to extend ZSL to very large scale settings, a significant performance gap still exists between prototypes obtained in an unsupervised way and with human annotations [22]. This gap will be further explored in Section 3.2.4.

One possible explanation for this gap is that the text corpora habitually used to train the word embedding models do not contain enough visual information. As illustrated in Figure 3.1, among the 20 most frequent words in the context of the word "tiger" in the Wikipedia corpus (with a context window of size 4), only "white" and "tail" could correspond to visual aspects of the word "tiger". Even

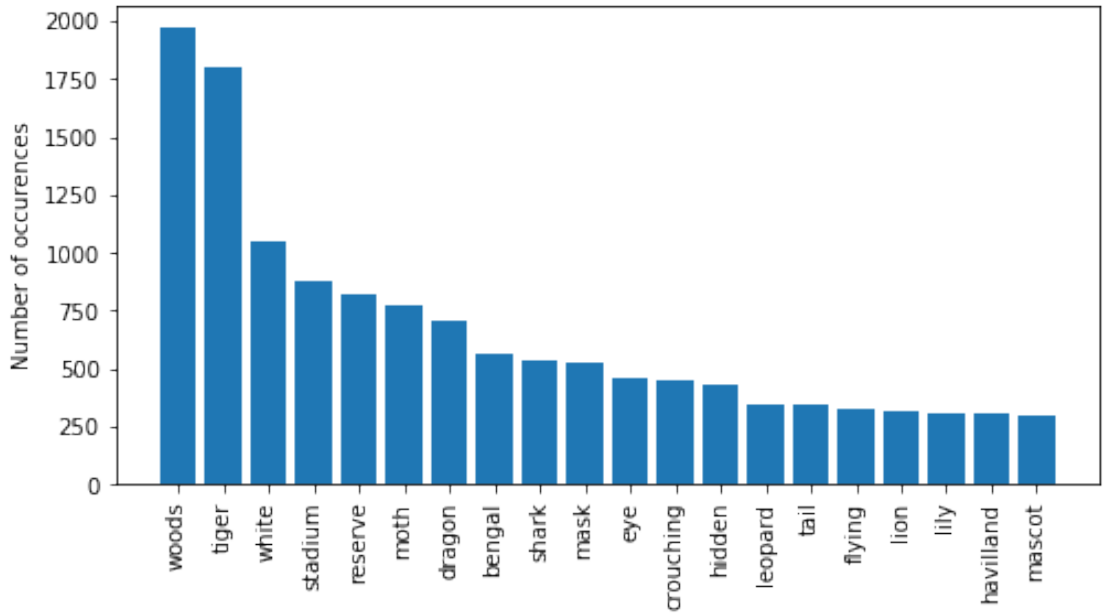

Figure 3.1 – Histogram of the most frequent words in a context window of size 4 around the word "tiger" in the Wikipedia corpus.

then, these words are not necessarily the most relevant, as white is not usually the predominant color of a tiger. We therefore do not expect the resulting embeddings to adequately encompass useful visual attributes of a tiger, such as "orange" or "stripes".

We can thus hypothesize that word embedding models trained on corpora with a more visual essence could lead to embeddings better suited for the ZSL task. We therefore propose to create such datasets.

3.1.1 Dataset collection

To ensure the datasets mostly contain words with a visual nature, we propose to collect corpora consisting of words or tags describing pictures. We use the Flickr API to collect tags defined by users: given a query q consisting of keywords such as *"tiger"*, the API returns a list of pictures and associated metadata. We are interested in three fields in the metadata associated with each picture:

- The title, which is a user-defined description of the image, for instance *"Amur tiger chilling in the water"*[2] for a result of the query *"tiger"*

[2]These examples were cherry-picked among results for illustrative purposes; for the query *"tiger"*, the first results frequently have the single word *"tiger"* as both the title and the tags.

- A list of user-defined tags associated with the picture. Examples include *"tiger", "sumatra", "wildlife"*[2] or *"tiger", "orange", "zoo"*[2] for the query *"tiger"*.

- The user identifier, so that each picture can be associated with a unique user. This will be employed in Section 3.1.2 to address the problem of *bulk-tagging*.

For each query, we keep the corresponding metadata associated with the first 5000 results. These results are sorted by "relevance" by the default Flickr ranking algorithm.

It is important to note that as all the content is generated by users, the titles and tags are not always relevant with respect to the pictures, and the pictures themselves do not necessarily correspond to the searched keywords. In addition, the annotations can be in any language. For instance, the following description and tags can be found among results corresponding to the query *"ivory gull"*: *"Ísmáfur Pagophila eburnea Ivory Gull"* and *"minnesota flying inflight gull arctic juvenile duluth rare lakesuperior canalpark ivorygull saintlouiscounty"*. Here, the title includes the Icelandic, Latin and English variants of the name while the tags themselves provide information about the location and activity of the ivory gull. Furthermore, tags can be single words (*"gull"*) or concatenated ones (*"ivorygull"*,*"lakesuperior"*).

We still need to determine the queries to be used to create the full corpus. As a first approach, we can use generic concepts from Wikipedia to create the Flickr-Wikipedia collection, or $\mathbf{fl_{wiki}}$. To select common generic concepts, Wikipedia pages are ranked by the number of associated incoming links in the Wikipedia corpus, and the top 120,000 concepts are kept. The corresponding titles of these pages are used as queries to collect metadata using Flickr as described previously. These 120,000 queries result in 62.7 million results (pictures from which we only keep the metadata), corresponding to a total of 1.11 billion words. This approach has the advantage of being agnostic to the types of classes used in the ZSL task: the dataset can be collected once, and the resulting word embeddings can then be employed for multiple ZSL datasets.

In a second approach called $\mathbf{fl_{cust}}$, we use the class names of the ZSL classes to collect metadata specifically suited to the task. This approach assumes that we have access to the names of both seen and unseen classes before training the ZSL model. As a result, we can consider that this approach operates in a class-transductive setting, and is more restrictive than the $\mathrm{fl_{wiki}}$ approach as such an

assumption is not always valid in real use-cases. However, as mentioned in previous chapters, this is still a frequent assumption in many ZSL works, particularly with generative models (Section 1.2.4). Combining the class names of the three datasets CUB, AwA2 and ImageNet[3] used in the evaluation section of this chapter (Section 3.2), this corresponds to 40,989 queries resulting in 61.9 million results for a total of 995 million words.

To summarize, each collection consists of Q concepts with associated queries $\mathcal{Q} = \{q_1, \dots, q_Q\}$, $Q \leq 120,000$. For each query q, we have a list of M_q results with associated metadata pieces $\mathcal{M}_q = \{m_1, \dots, m_{M_q}\}$, $M_q \leq 5,000$. Each metadata piece m consists of a list of words $\mathcal{W}_m = \{w_1, \dots, w_{T_m}\}$, as well as a user identifier id_m mapping the author id_m of the picture and metadata m with a unique user u_{id_m}. The T_m words are the words constituting the title and tags, with stopwords removed.

3.1.2 Corpus pre-processing

Since we have collections of words, we could consider applying the standard skip-gram objective (Equation (1.79)), either to each list of words $\{w_1, \dots, w_T\}$ or to concatenations of these lists, to learn word embeddings. However, such a direct approach raises several problems.

First, unlike in standard text collections such as Wikipedia, the order of the words $\mathcal{W}_m = \{w_1, \dots, w_{T_m}\}$ in each metadata result m is somehow arbitrary. Even though there can be some order in words in the titles, this is not always the case as illustrated previously. As for the order of the tags, it is completely meaningless. As a result, no specific order is preserved when the list of words $\mathcal{W}_m$ is extracted from the title and tags of m. Consequently, the fixed size context window from the usual skip-gram formulation (Equation (3.1)) is not well suited since two words appearing in the same context, e.g. in the same metadata result m, are not necessarily close to each other in $\{w_1, \dots, w_{T_m}\}$.

Instead of using a fixed size window, i.e. considering that two words appear in the same context if they are closer to each other than the size S of the context window in Equation (1.79), we consider that two words w_i and w_j appear in the same context if both of them appear in $\mathcal{W}_m$ of the same metadata result m. As a result, the more frequently two words w_i and w_j are used to describe the same picture, the stronger the semantic link between the two will be. The skip-gram objective in

[3]CUB and AwA2 have respectively 200 and 50 classes with a single class name per class; the full ImageNet dataset has a total of 21,842 classes with certain classes having several class names or lemmas, which results in a total of 40,739 concepts for this dataset.

Equation (3.1) can therefore be rewritten as

$$-\sum_{q\in\mathcal{Q}}\sum_{m\in\mathcal{M}_q}\sum_{\substack{(w_i,w_j)\\ w_i,w_j\in\mathcal{W}_m,\ i\neq j}} \log p(w_i|w_j) \tag{3.3}$$

This is equivalent to extracting all pairs of words (w_i, w_j) such that w_i, w_j belong to the same $\mathcal{W}_m$ similarly to Equation (3.3), and creating a corpus whose sentences consist of pairs of such words. Training the embedding model can then be achieved with the skip-gram objective from Equation (3.1). The same method can be applied to learn embeddings with GloVe [117] and FastText [14]. This has the advantage of enabling the use of available implementations for different word embedding models. The corpus created from all pairs of words appearing in the same context on which embedding models are trained will sometimes be referred to as the *training corpus*.

Another problem arising with datasets consisting of user-defined tags is sometimes referred to as *bulk-tagging* [111]: it describes the action of a user attributing the same tags or description to a whole set of photos. *Semi-bulk* is a related problem, where a few tags are attributed by a user to a set of photos and completed with picture-specific tags for each photo. These phenomena are known to bias language models obtained from Flickr [121, 111]. As an example, if a user posts many photos with the description or tags *"nice tiger"*, the words "nice" and "tiger" may become more related than we would intuitively expect.

To avoid this problem, we enforce the rule that a pair of words (w_i, w_j) is taken into account at most once per distinct Flickr user. This translates into adding a pair (w_i, w_j) in the training corpus only once for each user. Such an approach resulted in interesting performance gains in [121] and [111], on the respective tasks of image retrieval and automatic geo-tagging. It also has the positive side effect of decreasing the size of the training corpus, such that embeddings can be learned faster. The impact of restricting the importance of a single user will be evaluated in Section 3.2.3.

3.2 Evaluation of the proposed semantic embeddings

We compare results obtained with the proposed approach to results obtained with standard embeddings trained on generic text corpora, for a variety of ZSL models.

3.2.1 Experimental setting

Baseline methods. In the **wiki** approach, we train word embeddings with the standard methods Word2vec, GloVe and FastText on the customary Wikipedia corpus. This corpus contains well-formed sentences such as *"The ivory gull is found in the Arctic, in the northernmost parts of Europe and North America."*, and is often used to train embeddings for a variety of tasks [101, 102, 117] because it covers a wide array of topics [45]. We make use of a dump from January 2019 which includes 20.8 billion words. This is actually the same data as the one we used to obtain the 120,000 concepts for our method described in Section 3.1.1. However, as mentioned earlier, the Wikipedia corpus does not specifically describe visual relations between words.

In the **clue** approach, we train word embeddings on an subset of the ClueWeb12 [21, 20] collection, with the specific goal to extract more visually oriented textual content. The full collection consists of 733 million Web pages which were collected so as to cover a wide variety of topics. We extract data associated with the images referenced in the dataset, by retrieving metadata from the corresponding *title* and *alt* HTML attributes. The resulting content is usually quite similar to what we retrieved from Flickr earlier, and typically consists of short texts such as *"ivory gull flying"*. After sentence deduplication [101], the resulting corpus includes 628 million unique metadata pieces and 3.7 billion words.

Finally, for each of the three embedding models Word2vec, GloVe and FastText, we compare our approach to generic pre-trained embeddings similar to the ones habitually used in previous ZSL works [40, 22, 53]. We employ common pre-trained embeddings freely available on the Internet[4]. Word2vec was trained on the Google News corpus including 100 billion words, GloVe was trained on a Common Crawl version with 840 billion token, and FastText was trained on a Common Crawl version with 600 billion tokens.

[4]The pre-trained embeddings can be downloaded at the following URLs:
- Word2vec: `https://code.google.com/archive/p/word2vec/` (version trained on GoogleNews with 100 billion words)
- FastText: `https://fasttext.cc/docs/en/english-vectors.html` (version trained on Common Crawl with 600B tokens, no subword information).
- GloVe: `https://nlp.stanford.edu/projects/glove/` (version trained on Common Crawl with 840B tokens).

ZSL datasets. Since we are interested in results in a large scale setting, we mostly conduct our experiments on the ImageNet dataset [33]. It was proposed in [40] to use the 1,000 classes from ILSVRC [132] as seen training classes, and the remaining 20,841 as unseen test classes. However, as mentioned in Section 1.3.2, it has been recently shown that a structural bias appears in this setting which allows a "trivial model" to outperform most existing ZSL models [53]. For this reason, we adopt the evaluation protocol proposed by Hascoet *et al.* [53], who consider the same training classes as [40] but employ 500 classes with a minimal structural bias for testing. We use the same visual features as [53], which consist of features extracted with a ResNet-101 model pre-trained on ImageNet.

To get insight into the gap existing between manual attributes and unsupervised embeddings, we also conduct experiments on smaller benchmarks on which the ZSL task is usually conducted with manual attributes. We therefore also make use of the CUB [154] and AwA2 [161] datasets, described in Appendix A.1. The usual manual attributes of CUB and AwA2 are respectively 312 and 85-dimensional. In our setting, we are only concerned with semantic prototypes which can be obtained automatically; our results therefore cannot be directly compared to the state-of-the-art algorithms which exploit manual attributes. Apart from the attributes, we adopt the experimental protocol of Xian *et al.* [161], and specifically their "proposed splits", for these two datasets. This setting is very similar to the one we employed in Section 2.5, the only differences being the semantic representations.

ZSL methods. We conduct experiments with DeViSE [40], ESZSL [130] and ConSE [108] (described in Section 1.2) as they are the three standard methods used in Hascoet *et al.* [53], and therefore the only methods for which comparable results are currently available. Although results for other models – namely GCN-6 [158], GCN-2 and ADGPM [60] – are also reported in [53], these models are based on graph-convolutional networks [67] which make use of additional intermediate nodes in the WordNet [103] hierarchy. Such methods are outside the scope of this section. We additionally provide results for SynC [22] as well as $\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$ and $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$. We employ the same protocol as in Section 2.5 to train the models and select the hyper-parameters. For ImageNet, we sample 200 random classes as validation classes. Following Section 2.7.1, we report results averaged over 5 runs with different random initialization of the models parameters when applicable.

Implementation details. Word embeddings are computed with the original implementations of Word2vec [102], GloVe [117] and FastText[14], with the same hyperparameters[5]. In particular, we follow the usual text processing steps they propose, and employ an embedding dimension of $K = 300$ for all embedding models. Semantic prototypes for all classes are computed using the same protocol as [53] for fair comparison. For the same reason, we employ the implementation from [53] to run ConSE, ESZSL and DeViSE. We use the implementation from [22] for SynC, and our own implementation for $\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$ and $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$. All semantic prototypes are $\ell 2$-normalized except with ESZSL to have a setting similar to [53] when applicable.

3.2.2 Results

The main results for the large scale ImageNet dataset are reported in Table 3.1. We evaluate the three embedding models Word2vec, GloVe and FastText, trained with our two approaches fl_{wiki} and fl_{cust} as well as on the two corpora *wiki* and *clue*, and we report results for existing pre-trained embeddings.

The best results overall for each ZSL model are consistently obtained with the fl_{cust} approach with FastText. Class prototypes obtained with this approach enable to significantly outperform previous ZSL results in a large scale setting. In particular, the best previously reported result in this setting was 13.5 with ESZSL in [53]. By contrast, the best result in Table 3.1 is 17.2 with the simple $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$model with fl_{cust} and FastText, obtained with embeddings trained on a dataset more than 800 times smaller. The best result with ESZSL in Table 3.1 is 15.8 with fl_{cust} and FastText. It can also be emphasized that the absolute best previously reported result was 14.1 in [53] with the ADGPM [60] model, which additionally makes use of the WordNet hierarchy.

As a side note, results with ConSE, ESZSL and DeViSE with the GloVe pretrained embeddings (marked with "*" in Table 3.1) can be seen as a reproduction of the results from [53] since we employ the same visual features, and the GloVe embeddings pre-trained on Common Crawl are used as semantic embeddings in [53]. Our results are mostly consistent with the ones reported in [53], as we obtain

[5]The original implementations of each method are available at:
- Word2vec: `https://code.google.com/archive/p/word2vec/`
- Glove: `https://nlp.stanford.edu/software/GloVe-1.2.zip`
- FastText: `https://github.com/facebookresearch/fastText`

The hyperparameters as well as the corresponding commands used to train these models are provided respectively in Table A.1 and Table A.2 of Appendix A.2.

	pt	wiki	clue	$\mathbf{fl_{wiki}}$	$\mathbf{fl_{cust}}$
Word2vec					
ConSE	*9.9*	10.5	11.3	11.9	13.5
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*6.8*	9.8	9.6	10.5	12.6
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*11.6*	11.8	12.2	12.8	17.1
ESZSL	*10.5*	10.0	10.7	9.5	15.3
DeViSE	*9.0*	9.8	9.9	9.6	13.3
SynC	*12.2*	12.4	12.6	12.5	16.3
GloVe					
ConSE	*11.3**	8.1	7.8	11.3	11.9
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*10.2*	6.2	4.2	9.6	9.2
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*14.1*	7.9	8.0	9.2	11.4
ESZSL	*14.1**	8.0	10.3	11.1	12.0
DeViSE	*11.0**	5.9	5.4	3.8	3.4
SynC	*15.0*	10.9	11.2	12.4	13.3
FastText					
ConSE	*11.0*	10.5	5.4	12.6	**14.5**
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*6.0*	8.9	2.8	11.6	**14.2**
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*14.4*	12.1	8.0	13.3	**17.2**
ESZSL	*14.2*	10.1	1.1	11.9	**15.8**
DeViSE	*12.3*	10.1	5.6	10.3	**13.8**
SynC	*14.6*	12.6	7.0	13.2	**16.5**

Table 3.1 – ZSL accuracy on the large scale ImageNet dataset, for three embedding models Word2vec, GloVe and FastText. We compare the results from the proposed approaches fl_{wiki} fl_{cust} and to the baselines *wiki* and *clue* as well as pre-trained embeddings (*pt*). We use the experimental protocol from [53]. Results marked with "*" correspond to a setting close to Table 2 from Hascoet *et al.* [53], and are consistent with the results reported there.

	pt	wiki	clue	$\mathbf{fl_{wiki}}$	$\mathbf{fl_{cust}}$
Word2vec					
ConSE	*8.3*	19.5	21.6	18.0	21.1
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*7.5*	14.0	13.9	12.2	16.3
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*11.3*	18.0	17.2	21.5	23.0
ESZSL	*15.8*	20.4	17.9	23.0	25.2
DeViSE	*12.6*	17.0	15.8	19.0	19.2
SynC	*15.3*	19.8	17.3	20.3	21.3
GloVe					
ConSE	*14.1*	15.1	14.9	16.8	18.4
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*8.0*	11.6	9.8	12.7	14.2
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*18.2*	16.0	13.4	14.6	19.0
ESZSL	*19.9*	17.5	16.9	19.0	20.8
DeViSE	*14.6*	16.3	9.9	18.4	14.8
SynC	*17.6*	17.2	17.6	21.6	20.5
FastText					
ConSE	*14.0*	17.7	19.9	17.6	**23.4**
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*7.2*	13.8	12.2	11.6	**17.5**
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*16.1*	16.2	16.0	19.9	**24.4**
ESZSL	*21.1*	18.7	1.7	23.5	**26.5**
DeViSE	*16.0*	13.2	13.7	17.4	**22.5**
SynC	*17.0*	15.0	15.7	20.2	**24.0**

Table 3.2 – ZSL accuracy on the smaller scale CUB dataset with unsupervised semantic embeddings. We use the "proposed splits" from Xian *et al.* [163].

an accuracy of 11.3 with ConSE compared to 10.6 in [53], 14.1 compared to 13.5 with ESZSL, and 11.0 compared to 11.1 with DeViSE. Our slightly different reported accuracies may be attributed to small differences during the pre-processing of class names as well as random noise due to the different initializations and choice of validation classes.

We also provide results for the smaller scale datasets CUB and AwA2 in tables 3.2 and 3.3. These results are less relevant since manual attributes exist for these datasets, but still bring interesting insights. Importantly, these results are produced using *unsupervised* prototypes, which should be kept in mind when comparing to results obtained with manual attributes. On CUB, the best results are obtained with the embeddings learned on the fl_{cust} collection for the three configurations and significantly outperform previous embeddings. Interestingly, there does not seem to be a clear tendency on AwA2. It turns out that performance obtainable with unsupervised prototypes on AwA2 is already

	pt	wiki	clue	$\mathbf{fl_{wiki}}$	$\mathbf{fl_{cust}}$
Word2vec					
ConSE	*27.4*	31.3	34.3	**43.3**	39.2
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*31.1*	40.2	38.5	**43.6**	37.9
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*38.1*	44.1	49.7	53.9	55.0
ESZSL	*40.9*	42.2	55.8	53.1	57.1
DeViSE	*37.2*	34.1	46.6	33.7	43.4
SynC	*43.9*	41.1	45.8	47.1	47.5
GloVe					
ConSE	*31.3*	27.4	29.8	38.4	41.4
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*40.4*	26.9	34.6	40.5	43.3
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*56.6*	42.4	48.1	41.2	**57.7**
ESZSL	***61.4***	37.7	49.0	48.2	44.3
DeViSE	*43.2*	42.6	44.9	30.6	36.4
SynC	*46.9*	46.6	47.4	50.0	52.1
FastText					
ConSE	*34.7*	31.3	16.7	42.3	42.1
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	*42.1*	39.9	28.1	38.5	41.6
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	*54.7*	49.3	14.4	50.4	46.5
ESZSL	*48.2*	37.6	7.9	49.7	54.6
DeViSE	***52.0***	40.7	13.5	32.7	37.6
SynC	***53.3***	40.0	15.2	45.5	48.1

Table 3.3 – ZSL accuracy on the smaller scale AwA2 dataset with unsupervised semantic embeddings. We use the "proposed splits" from Xian *et al.* [163].

	Unique (w_i, w_j) per user	No restriction
ConSE	14.5	12.6
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	17.2	13.8
ESZSL	15.8	12.5
DeViSE	13.8	11.2

Table 3.4 – ZSL accuracy on the ImageNet dataset for different models with the fl_{cust} approach with FastText embeddings, with distinct pairs of words (w_i, w_j) limited to 1 per user *(left)*, or without restrictions on the impact of each user *(right)*.

quite close to performance with manual attributes – as detailed in Section 3.2.4. The proposed method is therefore unable to provide a significant improvement, unlike on the other two datasets.

3.2.3 Ablation of user filtering

In Section 3.1.2, we proposed to employ a simple mechanism to prevent a single user from having too much impact on the collected corpus, in order to minimize for instance the effect of bulk-tagging. This mechanism consists in using a pair of words (w_i, w_j) at most once per user when creating the training corpus. In order to measure the impact of this preprocessing step, we compare results obtained with and without this step with the approach fl_{cust} with the best performing embedding model FastText on the ImageNet dataset for different ZSL models. These results are reported in Table 3.4, showing a gain from 1.9 to 3.4 points of accuracy depending on the models. It confirms that limiting the impact of a single user on the training corpus has a significant positive impact on the final performance of the ZSL model.

3.2.4 Comparison to manual attributes

Although our webly semantic prototypes enable to achieve significantly better results than with previously available prototypes extracted from text corpora, it is still interesting to compare them to what can be achieved with hand-crafted attributes. Such attributes do not exist for very large scale datasets such as ImageNet, but they are provided with smaller scale datasets such as CUB and AwA2.

To quantify how much better hand-crafted prototypes perform when compared to webly supervised prototypes, we progressively remove attributes from the class prototypes of CUB and AwA2. We start with the full list of attributes, initially comprising 312 attributes for each bird species from CUB

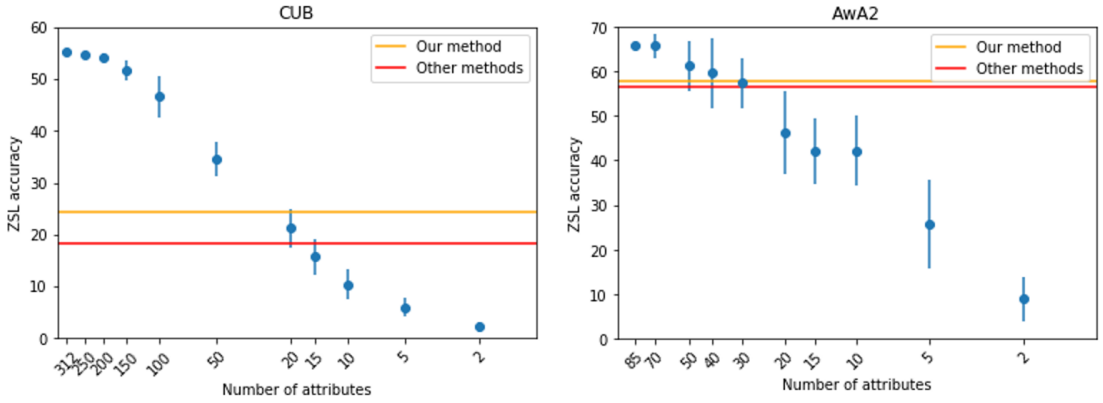

Figure 3.2 – Ablation of manual attributes on the CUB *(left)* and AwA2 *(right)* datasets. Each time, a random subset of the attributes is selected, and the resulting ZSL score is measured with the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model. The blue dots indicate the mean score over 10 runs with different random attributes selected, the vertical blue bars indicate corresponding standard deviations. Best results for prototypes based on unsupervised word embeddings are also reported for the proposed method (yellow horizontal line) and previous embeddings (red horizontal line), all with the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model.

and 85 attributes for each animal species from AwA2, and we randomly remove attributes while measuring the resulting ZSL score. The scores are measured for the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model due to its good results, robustness and simplicity. To account for the noise caused by the randomness of the removed attributes, each reported score is the average of 10 measurements, each with different random attributes removed. The remaining attributes are $\ell 2$-normalized, and the hyper-parameter λ (Equation (1.38)) is re-selected by cross-validation for each run. Figure 3.2 provides a visualization of the results, the blue dots representing the average score for a fixed number of hand-crafted attributes. We also display as a reference the results of the unsupervised embeddings with the highest scores for previous pre-trained embeddings as well as for our proposed method, also measured with the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model.

On CUB, there is still a substantial margin for improvement; even though the proposed approach enables a significant increase over other methods, the ZSL score is still barely above results achievable by selecting only 20 attributes among the 312 initial attributes. Interestingly, the difference between webly supervised and hand-crafted prototypes is not so pronounced on the AwA2 dataset; the ZSL accuracies of the two settings are even surprisingly close. This may be explained by the fact that AwA2 only contains 10 test classes; class prototypes need not enable a ZSL model to subtly distinguish very similar classes. Consequently, the best result of the proposed approach is comparable to the best result provided by previous methods.

Collection size		100%	50%	25%	10%
wiki	ConSE	10.5	11.0	10.5	9.9
	Ridge$_{\mathcal{S}\to\mathcal{V}}$	12.1	11.6	11.3	10.2
	ESZSL	10.1	9.8	9.9	9.6
	DeViSE	10.1	8.3	8.7	8.0
fl$_{\text{cust}}$	ConSE	14.5	14.1	14.1	14.3
	Ridge$_{\mathcal{S}\to\mathcal{V}}$	17.2	16.8	16.3	15.6
	ESZSL	15.8	15.1	15.3	14.3
	DeViSE	13.8	13.4	13.2	12.5

Table 3.5 – ZSL performance with 100%, 50%, 25% and 10% of the initial data from the *wiki* and *fl$_{cust}$* collections. Results obtained on the ImageNet dataset, with FastText embeddings.

3.2.5 Influence of collection size

The quality of semantic embeddings is usually influenced by the size of the text collections used to train the embedding models [102, 117]. To evaluate the effect of the collections sizes on our embeddings, we ablate 50%, 75% and 90% of the *wiki* and *fl$_{cust}$* collections and report results for ImageNet using the FastText embeddings in Table 3.5. Performance is as expected correlated to the collection size, with the best results being obtained for full text collections and the worst when 90% of them is removed. Interestingly, the performance drop is not drastic for either collection. For instance, with only 10% of the initial collections, accuracy decreases from 12.1 to 10.2 for *wiki* (a 15.7% relative decrease) and from 17.2 to 15.6 for *fl$_{cust}$* (a 9.3% relative decrease) with the best performing Ridge$_{\mathcal{S}\to\mathcal{V}}$ model. This is all the more surprising as the score of 15.6 obtained with only 10% of the *fl$_{cust}$* collection can still be considered to be state-of-the-art with respect to previously published results.

3.2.6 Error analysis

We analyze how far incorrect predictions are from the correct class by computing the distance between the predicted class and the correct class. We define the distance between two classes as the shortest path between them in the WordNet hierarchy. For a given distance d, we measure the number of predictions that are exactly d nodes away from the correct class, a distance of 0 being a correct prediction. Results for *wiki* and *fl$_{cust}$* are presented in Figure 3.3(a); the general tendency seems to be that classes farther away from the correct class are less likely to be predicted. Note that no two test classes are at a distance of one from each other, since it is not possible for a test class to be a direct

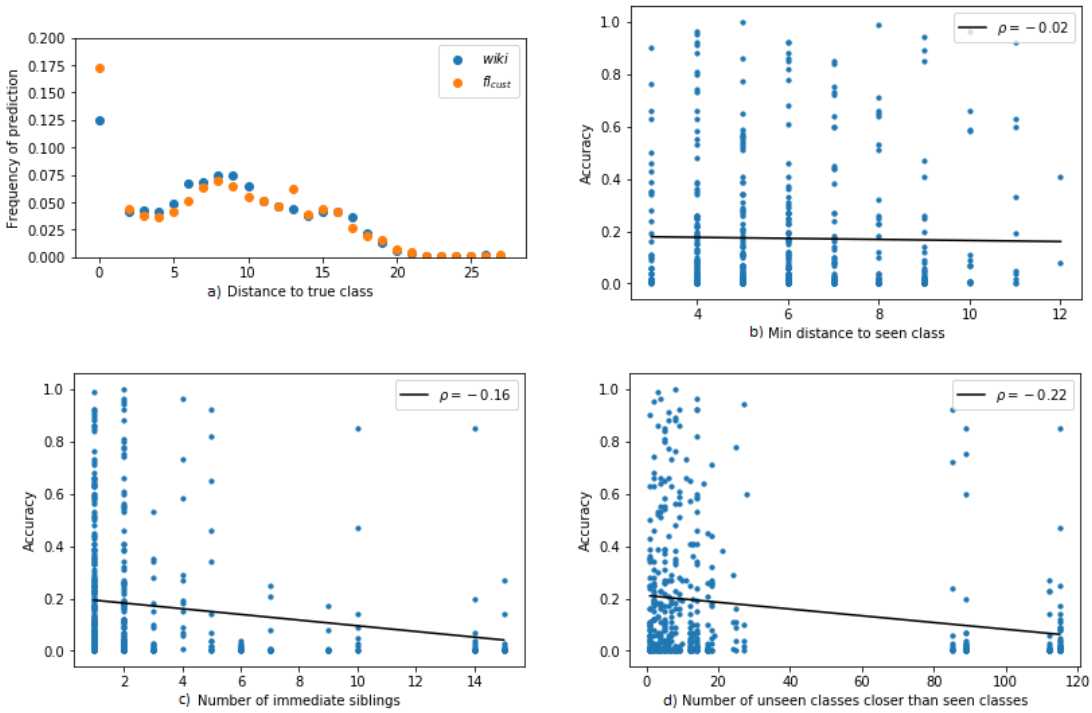

Figure 3.3 – (a) Distance from predicted class to correct class in the WordNet hierarchy. Correlation ρ between ZSL accuracy and (b) distance to the closest seen class, (c) the number of immediate unseen test class siblings, (d) the number of unseen classes closer than the closest seen class, for all 500 unseen ImageNet classes.

parent or child of another test class.

We further analyze the main factors behind classification errors. Experiments below are conducted on ImageNet, with the Ridge$_{\mathcal{S}\to\mathcal{V}}$ model trained using the FastText *fl*$_{cust}$ embeddings. A first hypothesis is that the distance between unseen and seen classes influences classification accuracy: the less an unseen class resembles any seen class, the harder it is to identify. To test this hypothesis, we consider for each unseen class c_u the minimal distance to a seen class $\min_{c\in\mathcal{C}_s} d(c_u, c)$, and analyze its relation to the prediction accuracy. The resulting plot is displayed in Figure 3.3(b). Surprisingly, the distance to the closest seen class seems to have little to no effect on the accuracy (correlation $\rho = -0.02$).

Another hypothesis may be that unseen classes close to other unseen classes are harder to classify than isolated unseen classes, as more confusions are possible. For each unseen class, we therefore compute the number of immediate siblings, a sibling being defined as an unseen class having the same parent in the WordNet hierarchy as the reference (unseen) class. The link between this metric and class accuracy is slightly stronger, with a correlation $\rho = -0.16$ as illustrated in Figure 3.3(c), but still weak overall.

We combine these two hypotheses by considering the number of unseen classes closer than the closest seen class for each unseen class. The link with class accuracy is more pronounced than by simply considering the number of siblings, with a correlation $\rho = -0.22$ as illustrated in Figure 3.3(d). Examples of classes at both ends of the spectrum are visible in Figure 3.4: unseen class *morel* (on the left) is close to seen class *agaric* and has no unseen siblings; its class accuracy is 0.63. On the other hand, classes *holly*, *teak* and *grevillea* (on the right) have many unseen siblings and are far from any seen class; their respective accuracy are 0.01, 0.00 and 0.03. More generally, classes which are descendant of the intermediate node *woody plant* have an average accuracy of 0.053. The full graph visualization of the 1000 training classes, 500 testing classes and intermediate nodes of the ImageNet ZSL dataset is available in Figure A.7.

3.3 Using sentences as semantic information

We have obtained interesting results in a large scale setting in Section 3.2, by creating and leveraging corpora with more visual components which enable to create better performing embeddings from class names. However, as evidenced in Figure 3.2, results with such an approach are still significantly

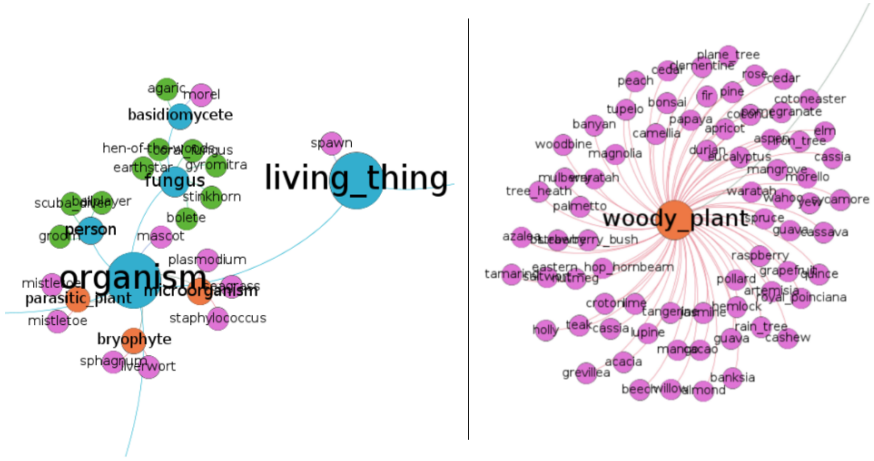

Figure 3.4 – Graph visualization of parts of the WordNet hierarchy. Green and pink leaves are resp. seen and unseen classes. Intermediate nodes are orange if there is no seen class among their children, and blue otherwise. Full graph is available in Figure A.7.

Figure 3.5 – Australian terrier *(left)* and Irish terrier *(right)*.

below those obtainable with human defined attributes on CUB. We argue that this approach may still be limited for fine-grained visual recognition. For instance, even with embeddings created from a more "visual" corpus, it is not obvious that it is possible to learn the difference between an *Australian terrier* and an *Irish terrier*, two classes illustrated in Figure 3.5 and present in the ImageNet dataset, just from their class name embeddings.

Even a human would certainly need more visual information to recognize these breeds, as the word *"terrier"* is common to the two classes, and the words *"Australian"* and *"Irish"* by themselves do not provide visual information. An ideal solution could be to use short natural sentences to describe each class, as this is less time-consuming than providing comprehensive attributes and can be more visually informative than word embeddings derived from generic text corpora. Examples of such short descriptions could be *"small greyish wire-haired breed of terrier from Australia"* and *"medium-sized breed with a wiry brown coat; developed in Ireland"* for the respective classes *Australian terrier* and *Irish terrier*. These examples are actually taken from the WordNet definitions [103] and correspond to the class descriptions we will employ in sections 3.3.1 and 3.3.2.

The use of short sentences as class descriptions in ZSL is not well studied. Although some works described in Section 1.3.2 make use of descriptions in natural language, they either employ large quantities of text extracted from the Wikipedia pages of each class [35, 89, 122, 36, 170], or they use 10 short sentences *per image* [126]. In both cases, providing these descriptions does not save human effort when compared to providing attributes. [54] experiments with different standard methods [68, 29] to obtain sentence representations from WordNet definitions for ZSL. However, reported scores are significantly below those obtained with word embeddings in the same article[6].

We therefore propose different approaches to employ short descriptions in natural language for zero-shot learning. A first approach consists in using a weighted average of word embeddings from a definition to build a usual single semantic prototype for a given class (Section 3.3.1). Another approach consists in adapting a ZSL model to make it capable of exploiting a variable number of prototypes per class, and using the word embeddings from a definition as a set of prototypes (Section 3.3.2). Both approaches are evaluated in Section 3.4.

[6]Experiments from [54] are performed on the biased version of ImageNet as described in [53] and in sections 1.3.2 and A.1. As a consequence, the results cannot be directly compared to ours and are therefore not reported.

3.3.1 Attention approaches

The most straightforward approach to obtain a semantic prototype from a short sentence description consists in averaging the embeddings of the words in the description, as is usually done for class names consisting of several words: if a sentence s describing a class has T words with respective embeddings $\{\mathbf{v}_1, \dots, \mathbf{v}_T\}$, then the corresponding semantic representation is

$$\mathbf{s} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{v}_t \tag{3.4}$$

We call this baseline the $\textbf{Def}_{\textbf{average}}$ approach. However, as illustrated in Figure 3.8, not all words are equally important in a short sentence description. We therefore explore the use of attention mechanisms: the sentence embedding is a weighted average of the embeddings of its words, so that more important words contribute more to the resulting embedding. We consider two ways to achieve this: an approach in which words are weighted by their "visualness" estimated from external data sources, and an approach in which the weights corresponding to each word are directly estimated from the word embeddings.

Unless stated otherwise, we employ the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model to evaluate the resulting class prototypes, as it has the best results on average in a large scale setting in Section 3.2 and is fairly simple with a single hyper-parameter λ (Equation (3.9)).

3.3.1.1 Visualness-based method

In the $\textbf{Def}_{\textbf{visualness}}$ approach, we aim to estimate how "visual" a word is. As an example, we intuitively expect the word *"striped"* to carry a more visual connotation than the word *"constitutional"*. For a given word w_i, we thus collect the $M_i \leq 100$ most relevant images from Flickr using the website's search ranking. Similarly to sections 1.3.1 and 2.5, we obtain visual representations $\{\mathbf{r}_1^i, \dots, \mathbf{r}_M^i\}$ for the M collected images using a pre-trained ResNet-101, such that $\mathbf{r}_m \in \mathbb{R}^{2048}$. We hypothesize that for words with high visual content, the visual representations of collected images corresponding to this word are close to each other. We thus measure the average distance of vectors $\mathbf{r}_m^i$ to the mean vector $\overline{\mathbf{r}}^i$ to obtain the quantity v_i:

$$\overline{\mathbf{r}}^i = \frac{1}{M} \sum_{m=1}^{M} \mathbf{r}_m^i \tag{3.5}$$

Figure 3.6 – *Top*: words with highest visualness. *Bottom*: words with lowest visualness. The visualness of a word is the inverse of the mean distance (shown in parenthesis) to the mean representation of visual features from the top 100 corresponding images from Flickr. Top 1 image with no copyright restriction is displayed. Words with the highest and lowest visualness as well as corresponding inverse visualness (the mean distance to the mean feature representations for images associated with this word) and the corresponding top image result with no copyright restriction from Flickr.

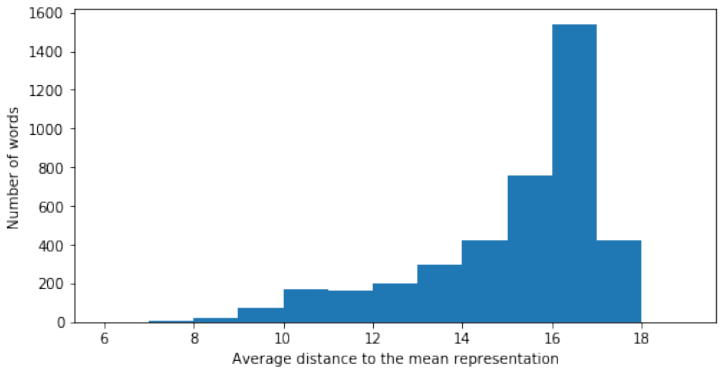

Figure 3.7 – Inverse of the visualness (low values correspond to high visualness) for the 4059 words from class names and WordNet definitions.

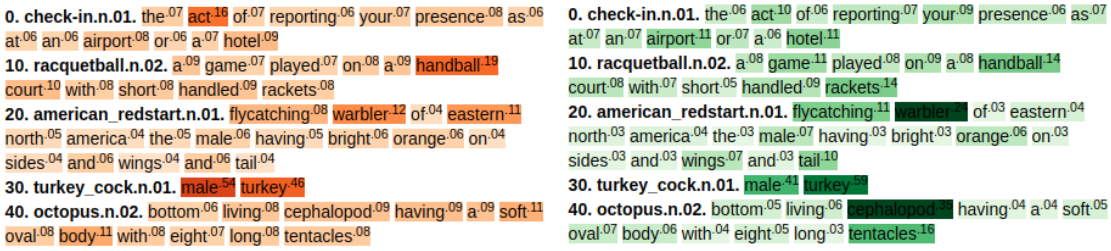

Figure 3.8 – Illustration of definitions and attention scores on some test classes from ImageNet, with the associated WordNet definitions. *Left*: weights from the $Def_{visualness}$ approach after softmax; the temperature is $\tau = 5$ so differences are less pronounced than initially. *Right*: weights learned with the $Def_{attention}$ approach, with FastText embeddings.

$$v^i = -\frac{1}{M}\sum_{m=1}^{M} \|\mathbf{r}_m^i - \bar{\mathbf{r}}^i\|_2 \tag{3.6}$$

which can be interpreted as the "visualness" of word w_i, i.e. a measure of how visual the concept associated with w_i is. Similarly to outliers in Section 2.3 being far from more representative samples, our intuition is that concepts with low visual content have associated images consisting mostly of outliers, and thus have a large average inter-class distance. Examples of words with high and low visualness shown in Figure 3.6 tend to confirm that this hypothesis is reasonable. Figure 3.7 shows the distribution of the additive inverse of the visualness v_i – i.e. the average of the raw distances to the class mean vector $\bar{\mathbf{r}}^i$ for each word w_i – for the 4059 unique words from the classnames and WordNet definitions of the 1000 training classes and 500 testing classes from [53] that we used in Section 3.2.

Given a definition with T words with corresponding embeddings $\{\mathbf{v}_1, \ldots, \mathbf{v}_T\}$, we then apply a softmax on the corresponding visualnesses $\{v_1, \ldots, v_T\}$ to obtain a sentence representation $\mathbf{s}$ from the weighted average of the embeddings, giving more weight to visual words. As the initial scale of the average distances / negative visualnesses is arbitrary, a temperature τ is introduced in the softmax, so that the resulting sentence embedding is

$$\mathbf{s} = \sum_{t=1}^{T} \frac{\exp(v_t/\tau)}{\sum_{k=1}^{T}\exp(v_k/\tau)}\mathbf{v}_t \tag{3.7}$$

τ is considered to be a hyper-parameter and its value is selected by cross-validation on the validation set, as detailed in Section 3.4.1. In practice, this often leads to selecting $\tau = 5$. An illustration of the resulting weights for a few sentences is shown in Figure 3.8 (left).

3.3.1.2 Learned attention

In the $\textbf{Def}_{\textbf{attention}}$ approach, we aim to learn to predict the visualness v^i of word w^i from its embedding $\mathbf{v}^i \in \mathbb{R}^K$ such that

$$v^i = \mathbf{w}^\top \mathbf{v}^i \tag{3.8}$$

where $\mathbf{w}$ are learned parameters. Equation (3.7) can then be used to create a prototype from a class definition. As the visualnesses v^i are directly learned, it is no longer necessary to account for their initial scale. We can thus discard the temperature in the softmax by setting $\tau = 1$ in Equation (3.7).

Different ways could be considered to learn the parameters $\mathbf{w}$ from Equation (3.8). A straightforward approach could consist in randomly initializing $\mathbf{w}$ along with the parameters $\mathbf{W}$ from the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model (Equation (3.9)), then computing visualnesses $v^i = \mathbf{w}^\top \mathbf{v}^i$ for each word w^i, computing class prototypes $\mathbf{s}_c$ for each class c using the v^i and Equation (3.7), computing $\mathbf{T} = (\mathbf{t}_1, \ldots, \mathbf{t}_N)^\top$ with $\mathbf{t}_n = \mathbf{s}_{y_n}$ similarly to Section 1.2.2, and finally computing the loss of the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model:

$$\frac{1}{N}\|\mathbf{X} - \mathbf{T}\mathbf{W}\|_2^2 + \lambda\|\mathbf{W}\|_2^2 \tag{3.9}$$

We could then use back-propagation and gradient descent to update $\mathbf{w}$ and $\mathbf{W}$ until convergence.

However, we instead take advantage of the existence of a closed-form solution for Equation (3.9) and proceed as follows: we randomly initialize $\mathbf{w}$ and compute class prototypes $\mathbf{s}_c$ and $\mathbf{T}$ using Equation (3.7) as previously. We then directly estimate $\mathbf{W}$ using the closed-form solution

$$\mathbf{W} = (\mathbf{T}^\top\mathbf{T} + \lambda N\mathbf{I}_K)^{-1}\mathbf{T}^\top\mathbf{X} \tag{3.10}$$

derived in Section 1.2.2 and employ this value to compute the loss in Equation (3.9). We then back-propagate the gradient and perform gradient descent on $\mathbf{w}$ only, the value of $\mathbf{W}$ being estimated with Equation (3.10) at each iteration. We repeat this process for 50 "epochs". An illustration of the resulting attention weights for a few sentences is shown in Figure 3.8 (right).

3.3.2 Multi-prototype approach

Instead of combining embeddings of words in a definition to form a single class prototype with a fixed dimension, another solution may be to adapt existing ZSL approaches to enable the use of a class prototype consisting of several parts with the same fixed dimension. We propose to explore this approach by adapting the triplet loss in order to accommodate several prototypes per class.

In the standard approach described in Section 1.2.3 and upgraded in sections 2.1 and 2.2, for a triplet $(\mathbf{x}, \mathbf{s}_c, \mathbf{s}_y)$ where $\mathbf{x} \in \mathbb{R}^D$ is a (training) visual sample with label y, $\mathbf{s}_y \in \mathbb{R}^K$ is the corresponding semantic prototype and $\mathbf{s}_c \in \mathbb{R}^K$ is a different prototype, the triplet loss takes the form

$$[m + f(\mathbf{x}, \mathbf{s}_c) - f(\mathbf{x}, \mathbf{s}_y)]_+ \tag{3.11}$$

Instead of having a class c represented by a unique semantic vector $\mathbf{s}_c$, we now consider that each class c is represented by a set of T_c prototypes $\{\mathbf{s}_c^1, \dots, \mathbf{s}_c^{T_c}\}$, with $\mathbf{s}_c^t \in \mathbb{R}^K$. Such prototypes may for example be the embeddings $\{\mathbf{v}_1, \dots, \mathbf{v}_{T_c}\}$ of the words constituting a short sentence description of the class.

The triplet loss in Equation (3.11) can then be adapted so that for instance, for the correct class y, only the textual embedding $\mathbf{s}_y^t$ with the highest compatibility $f(\mathbf{x}, \mathbf{s}_y^t)$ is taken into account, so that the triplet loss becomes

$$[m + f(\mathbf{x}, \mathbf{s}_c) - \max_{t \in [1, T_y]} f(\mathbf{x}, \mathbf{s}_y^t)]_+ \tag{3.12}$$

We note that in this example, we still consider that there is a unique prototype $\mathbf{s}_c$ for incorrect classes $c \neq y$ instead of a set of textual embeddings $\{\mathbf{s}_c^1, \dots, \mathbf{s}_c^T\}$. Such a prototype can be simply obtained by averaging the word embeddings such that $\mathbf{s}_c = \frac{1}{T} \sum_{t=1}^{T} \mathbf{s}_c^t$, similarly to the $Def_{average}$ approach from Section 3.3.1.

More generally, we can consider the P embeddings $\{\mathbf{s}_c^{p_1}, \dots, \mathbf{s}_c^{p_P}\}$ with the highest compatibility $f(\mathbf{x}, \mathbf{s}_y^p)$. Writing $\mathcal{T}_P = \{p_1, \dots, p_P\}$ the indexes of these "top-P" embeddings, the triplet loss becomes

$$[m + f(\mathbf{x}, \mathbf{s}_c) - \frac{1}{P} \sum_{p \in \mathcal{T}_P} f(\mathbf{x}, \mathbf{s}_y^p)]_+ \tag{3.13}$$

Finally, this approach can be extended so that we similarly consider the top-Q embeddings with the highest compatibility for incorrect classes $c \neq y$. Adapting the triplet loss from Equation (3.13) and summing the losses over the training set in a setting similar to DeViSE (Section 1.2.3), this results in the following total training loss:

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{\substack{c \in \mathcal{C}^{\mathcal{S}} \\ c \neq y_n}} \left[m + \frac{1}{Q} \sum_{q \in \mathcal{T}_Q} f(\mathbf{x}_n, \mathbf{s}_c^q) - \frac{1}{P} \sum_{p \in \mathcal{T}_P} f(\mathbf{x}_n, \mathbf{s}_{y_n}^p) \right]_+ \tag{3.14}$$

We note that if P and Q are equal to the number of words in each definition, this is equivalent to

	Word2vec	GloVe	FastText	Elmo
Classname	12.4	14.5*	14.8	10.9
$Def_{average}$	9.7	10.0	10.6	8.7
$Def_{visualness}$	10.5	10.5	10.9	9.5
$Def_{attention}$	10.5	10.2	11.0	9.5

Table 3.6 – Comparison of approaches on ImageNet with WordNet definitions, with the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model. The result marked with * corresponds to a setting similar to [53] (use of *Classname* with GloVe embeddings) but with a different model.

averaging the compatibilities with all the textual embeddings. If f is further linear in $\mathbf{s}$, this reduces to the standard triplet loss from Equation (3.11).

To make predictions with a model trained with the loss from Equation (3.14), we can again use the mean of the top R textual embeddings with the highest compatibility, and predict the unseen class c with the highest such mean:

$$\hat{y} = \underset{c\in\mathcal{C}^{\mathcal{U}}}{\text{argmax}} \frac{1}{R} \sum_{r\in\mathcal{T}_R} f(\mathbf{x}, \mathbf{s}_c^r) \tag{3.15}$$

Alternatively, we can make standard predictions similarly to Equation (1.4) from Section 1.1.3 using the mean of textual embeddings as a single prototype:

$$\hat{y} = \underset{c\in\mathcal{C}^{\mathcal{U}}}{\text{argmax}}\ f(\mathbf{x}, \frac{1}{R} \sum_{r\in\mathcal{T}_R} \mathbf{s}_c^r) \tag{3.16}$$

Both approaches from equations (3.15) and (3.16) are equivalent when the compatibility function f is linear with respect to $\mathbf{s}$.

3.4 Evaluation of sentence-based approaches

3.4.1 Evaluation of the visualness-based methods

The sentence embeddings obtained from the two approaches $Def_{visualness}$ and $Def_{attention}$ can be compared to the standard class prototypes obtained by embedding the class names, which we call the **Classname** approach. We conduct our experiments on the large scale ImageNet [33] dataset, using the WordNet [103] definitions corresponding to the synsets associated with the classes as short sentence descriptions. As stated in Section 3.3.1, we employ the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model to measure the ZSL performance associated with the obtained sentence embeddings.

We use the same experimental protocol as in Section 3.2, i.e. the same as in [53]. In particular, we employ the same train / test splits, and the same $\ell 2$-normalized visual features extracted with a pre-trained ResNet model. Similarly to Section 3.2, we evaluate performance with the three embedding models Word2vec, GloVe and FastText. We also conduct experiments with the Elmo [118] embedding model, which enables to obtain word embeddings that depend on the context of the words. This ability may be beneficial since we are interested in obtaining semantic representations from full sentences.

For Glove and FastText, we use the same pre-trained embeddings[4] as in Section 3.2.1; for Word2vec, we employ a model pre-trained on Wikipedia[7]. All three models have an embedding dimension of $K = 300$. For Elmo, we similarly use a pre-trained version[8]. Elmo embeddings have dimension 3×1024 (a 1024-dimensional embedding from each of the three layers). For the sake of simplicity we combine the three layers using the same weight of 0.33 to obtain a single 1024-dimensional representation for each word, as weights fine-tuned for our specific task gave similar results.

Results are shown in Table 3.6. The result for GloVe embeddings with the *Classname* approach (marked with "*") corresponds to a setting similar to the one from [53], but with the better performing $\text{Ridge}_{\mathcal{S} \to \mathcal{V}}$ model. Results with the *Classname* approach correspond to the results with pre-trained embedding models from Table 3.1.

The baseline $Def_{average}$ performs poorly compared to the usual *Classname* approach. Attention mechanisms provide a slight improvement, with comparable results for the two approaches $Def_{visualness}$ and $Def_{attention}$, but performance remains significantly below that of the *Classname* approach. This is consistent with results reported in [54], in which embeddings obtained from WordNet definitions also performed significantly worse than class name embeddings[6]. Similarly to Section 3.2, the embedding model has an impact on performance, with FastText performing consistently better than the other two non-contextual embedding models. Elmo has surprisingly low performance, including with attention, even though one could expect attention to be more effective here as Elmo can be considered to embed the role of a word in a sentence.

[7]The pre-trained embeddings for Word2vec can be downloaded from `https://wikipedia2vec.github.io/wikipedia2vec/pretrained/` (English version trained on Wikipedia with 300 dimensions).

[8]The pre-trained Elmo model can be downloaded from `https://allennlp.org/elmo`. We use the original version with 93.6 million parameters, pre-trained on the 1 Billion Word Benchmark [26].

	Multi-proto pred.	Standard pred.	$P = 1$	$P = \infty$
Multi-prototypes from lemmas				
DeViSE, no-norm	(11.0)	*11.0**	-	-
DeViSE, norm	(12.1)	*13.4*	-	-
Multi-P., no-norm	13.3	*11.6*	13.3	10.7
Multi-P., norm	13.5	*13.1*	13.5	12.0
Multi-prototypes from definitions				
DeViSE, no-norm	(6.3)	*6.3*	-	-
DeViSE, norm	(5.3)	*9.3*	-	-
Multi-P., no-norm	8.0	*3.8*	6.3	6.1
Multi-P., norm	8.0	*5.0*	5.3	5.1

Table 3.7 – *Multi-proto pred. column*: the Multi-Prototype model is trained with Equation (3.14). DeViSE is trained with the standard triplet loss similarly to Equation (3.11). Predictions are made with Equation (3.15), $P = Q = R$ is cross-validated when applicable. *Standard pred. column*: same as leftmost column, but predictions are made with Equation (3.16). $P = 1$ *column*: we fix $P = Q = R = 1$. $P = \infty$ *column*: all lemmas or all words from definitions are used. The results are obtained on the ImageNet dataset with WordNet definitions and lemmas, and GloVe embeddings.

3.4.2 Multi-prototype

We conduct experiments on the ImageNet dataset similarly to Section 3.4.1, with the same experimental setting. We compare results obtained with the multi-prototype approach from Equation (3.14) with a standard triplet loss approach (Equation (3.11)) similar to DeViSE, in which the word embeddings are averaged to obtain a single prototype $\mathbf{s}_c = \frac{1}{T}\sum_{t=1}^{T} \mathbf{s}_c^t$ per class c. We also compare with results obtained using a multi-prototype approach with class names instead of descriptions in natural language. Since some classes from ImageNet have "synsets" consisting of multiple lemmas, themselves possibly consisting of multiple words, we treat the different words from these lemmas in the same way as a short sentence.

For both the Multi-Prototype and the DeViSE model, we consider two variants: the "norm" variant, in which the projections $\mathbf{Ws}$ of semantic prototypes are $\ell 2$-normalized, so that the compatibility function is

$$f(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \frac{\mathbf{Ws}}{\|\mathbf{Ws}\|_2} \tag{3.17}$$

and the "no-norm" variant, in which the compatibility function is simply $f(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \mathbf{Ws}$. In the latter case, f is linear in $\mathbf{s}$. For both the DeViSE-like model and the Multi-Prototype approach, we use a margin of $m = 0.1$. For the sake of simplicity, we employ $P = Q = R$ in equations (3.14), (3.15)

1. pika (15) - small[2] short eared[24] burrowing[50] mammal[23] of rocky uplands of asia and western[6] north america
2. **mole** (15) - small[2] velvety furred[16] burrowing[50] mammal[23] having small[2] eyes and fossorial[18] forefeet
3. **staphylococcus** (14) - spherical gram[3] positive[3] parasitic[17] bacteria[5] that tend to form irregular colonies[34] some cause boils[3] or septicemia[23] or infections[26]
4. **brood_hen** (13) - a domestic[27] hen[7] ready to brood[39]
5. **plasmodium** (12) - parasitic[17] protozoan[25] of the genus[5] plasmodium[12] that causes malaria[21] in humans[2]

1. **pumpkin** (22) - a coarse vine[29] widely cultivated[1] for its large pulpy[25] round[20] orange[39] fruit[49] with firm orange[39] skin[2] and numerous seeds[31]
2. **morello** (21) - any of[0] several cultivated[1] sour[26] cherry[43] trees bearing fruit[49] with dark skin[2] and juice[29]
3. **grapefruit** (21) - citrus[39] tree[10] bearing large round[20] edible[18] fruit[49] having a thick yellow[26] rind[28] and juicy[33] somewhat acid pulp[8]
4. **quince** (21) - small asian tree[10] with pinkish[20] flowers[12] and pear[49] shaped fruit[49] widely cultivated[1]
5. **apricot** (21) - asian tree[10] having clusters of[0] usually white blossoms[16] and edible[18] fruit[49] resembling the peach[50]

...

55. tangerine (14) - a variety[7] of[0] mandarin[14] orange[39]

Figure 3.9 – Illustration of the word compatibilies associated with the descriptions of the top-5 candidates with the multi-prototype method. Compatibility is displayed for words with a positive compatibility only. $P = Q = R = 3$ is used for training and predictions. The correct class is displayed in orange.

and (3.16) when applicable. The exact value of this hyper-parameter is selected by cross-validation.

Results are shown in Table 3.7. The leftmost column *Multi-proto pred.* provides results with the model trained with the multi-prototype loss from Equation (3.14) (or a loss derived from Equation (3.11) for DeViSE), and predictions made with Equation (3.15) for both DeViSE (in which case we use all multi-prototypes to make predictions) and the Multi-Prototype model. The Multi-Prototype model obtains better results than DeViSE when we employ the distinct lemmas of a synset as multi-prototypes. Interestingly, a value of $P = 1$ is selected by cross-validation in this case, meaning that the model only considers the most compatible lemma. However, the results of the Multi-Prototype model with multi-prototypes consisting of words from definitions are still significantly below. In this case, a value of $P = 3$ is selected by cross-validation, meaning that the model considers the 3 most compatible words from a definition.

In the *Standard pred.* column, we use the same models as is the *Multi-proto pred.* column but we make "standard" predictions with Equation (3.16), using a single average prototype per class. Unsurprisingly, this results in better scores for DeViSE and worse scores for the Multi-Prototype approach, since the former was not trained with multiple prototypes while the latter was. We note that as expected, results are the same as in the *Multi-proto pred.* column for the "no-norm" variants of DeViSE since in these cases the compatibility function is linear in $\mathbf{s}$. These settings are similar to the one from [53], and we obtain comparable results with DeViSE using the embeddings from lemmas.

Finally, we conduct experiments setting P to 1 and to ∞, meaning in the latter case that we consider all multi-prototypes in Equation (3.14) (up to 9 lemmas and up to 45 words in definitions). The results are visible in the two rightmost columns of Table 3.7. For the multi-prototypes from lemmas, the results are the same as in the leftmost column since we selected a value of $P = 1$ by cross-validation. For the multi-prototypes from definitions, results are worse than in the leftmost column corresponding to $P = 3$ .This means that we selected a somehow relevant value of P, as using a single word or all the words leads to worse results in this context.

However, in all cases, the performance using definitions remains far below the one obtained with lemma embeddings, despite compatibility scores on individual words looking mostly reasonable (Figure 3.9). The performance reached with this approach is also below that of the attention approaches from Section 3.3.1.

3.5 Combination of sentences and class names

So far, the results obtained using short sentence definitions alone are not as good as the results obtained with standard unsupervised prototypes using class names in the form of lemmas. This may be partly explained by the fact that some of the definitions from WordNet do not really include relevant information to describe a class. For instance, for the two test classes *turtledove* and *Australian turtledove*, the corresponding descriptions are *"any of several Old World wild doves"* and *"small Australian dove"*. However, we hypothesize that in other cases there may still be some interesting additional information in these definitions compared to the class names alone.

We therefore experiment with a very simple approach to combine these sources of information: given $\mathbf{s}_1$ and $\mathbf{s}_2 \in \mathbb{R}^K$ two semantic representations obtained with different approaches, for example the *Classname* and $Def_{average}$ approaches from Section 3.3.1, we create a combined prototype $\mathbf{s}$ as a convex combination of the two prototypes parameterized by a scalar $\mu \in [0, 1]$:

$$\mathbf{s} = \mu\mathbf{s}_1 + (1 - \mu)\mathbf{s}_2 \tag{3.18}$$

We use this idea to combine *Classname* prototypes with representations of sentences from Section 3.3.1 which, contrary to the multi-prototype approaches, have the advantage of easily providing a single fixed-dimension representation of a sentence, and offer better performance. We call for instance **Classname+Def$_{\text{visualness}}$** the combination of the *Classname* prototype with the $Def_{visualness}$ prototype using Equation (3.18). Since recent results show that hierarchical and graph relations between classes contain valuable information [53, 60], in the **Classname+Parent** approach we combine the prototype obtained using a class lemmas with the prototype resulting from the lemmas of its parent in the WordNet hierarchy. We similarly define the $Classname{+}Def_{visualness}{+}Parent$ as the combination of the $Classname{+}Def_{visualness}$ prototype with the prototype from its parent class, the latter also being obtained by a combination of lemmas and definition embeddings.

Apart from the combination of prototypes, we use the same experimental protocol as in Section 3.4.1, and in particular the same train/test splits, visual features and word embedding models, and employ the Ridge$_{\mathcal{S}\to\mathcal{V}}$ model as the ZSL model unless otherwise specified. All prototypes are $\ell 2$-normalized before being combined. We select the value of μ jointly with the model hyper-parameters using cross-validation, except when combining a prototype with the prototype from its parent class.

	Word2vec	Glove	FastText	Elmo
Classname	12.4	14.5*	14.8	10.9
Classname+Parent	13.4	15.4	15.9	11.4
$Def_{average}$	9.7	10.0	10.6	8.7
$Def_{visualness}$	10.5	10.5	10.9	9.5
Classname+$Def_{average}$	14.6	16.9	17.2	12.2
Classname+$Def_{visualness}$	14.8	16.8	17.3	12.1
Classname+$Def_{visualness}$*+Parent*	**15.4**	**17.3**	**17.8**	**12.5**

Table 3.8 – Comparison of approaches on ImageNet with WordNet definitions, with the $\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$ model. The result marked with * corresponds to a setting similar to [53] (use of *Classname* with GloVe embeddings) but with a different model.

In the latter case, cross-validation tended to yield very inconsistent and unstable values. This is consistent with the findings of [4], which states that it was critical in their approach to cross-validate the weights on unseen classes when combining different semantic prototypes. In our case, whether validation classes can be considered truly unseen is debatable, as all the 1000 ILSVRC [132] classes we use as seen classes and unseen validation classes were used to train the visual feature extractor (Section 1.3.1). As a result, when combining a prototype with its parent prototype, we somehow arbitrarily fix $\mu = 0.75$, meaning the resulting prototype is 75% the child prototype and 25% the parent prototype.

Results are provided in Table 3.8. The combination of *Classname* with the *Def* approaches brings significantly better scores than either separately. Surprisingly, while any *Def* approach alone has lower performance than *Classname* alone, the best trade-off between the two in Equation (3.18) as determined by μ and selected by cross-validation consists in using 70% definition and 30% classname in every case. This means that counter-intuitively, the definition has a stronger presence than the class name in the resulting embedding.

The use of parent information in addition to the *Classname* prototypes improves results when compared to the child prototypes alone, which is consistent with [53] where the best methods make use of hierarchical relations between classes. The same effect is observed when using parent information with the *Classname+*$Def_{visualness}$ approach. This enables to reach a score of 17.8 with FastText embeddings, significantly higher than the 14.8 achieved with the use of *Classname* alone.

Finally, we provide additional results for different ZSL models in Table 3.9 with the best performing

	Top-1	Top-5	Top-10
ConSE*	10.6	25.1	-
ESZSL*	13.5	32.6	-
DeViSE*	11.1	29.5	-
GCN-6*†	9.6	27.2	-
GCN-2*†	14.1	35.1	-
ADGPM*†	14.1	36.0	-
ConSE	12.7	31.8	42.4
$\text{Ridge}_{\mathcal{V}\to\mathcal{S}}$	9.1	26.2	36.7
$\text{Ridge}_{\mathcal{S}\to\mathcal{V}}$	**17.8**	**43.6**	**56.7**
ESZSL	16.3	40.6	52.4
DeViSE	14.0	38.3	52.1

Table 3.9 – Top-k ZSL accuracy for different models, using the *Classname+Def*$_{visualness}$*+Parent* prototypes built from FastText embeddings. Results for models marked with * are reported from [53] and employ *Classname* prototypes with GloVe embeddings, but make use of additional graph relations for models marked with †.

approach, the *Classname+Def*$_{visualness}$*+Parent* approach with FastText embeddings, and report results from [53] for comparison. Performance is significantly improved for all models.

3.6 Discussion

We argued that the quality of semantic representations plays an important role in the performance of ZSL models, particularly in a large scale scenario. In this scenario, the usual word embeddings may be inadequate for fine grained recognition. We proposed several approaches to produce more suitable class prototypes, such as training embeddings on more visually oriented text corpora, or using a combination of class name embeddings and short descriptions in natural language as semantic representations. These methods enable to improve performance in a large scale setting while keeping the required human annotation effort to a reasonable level.

Similarly to Chapter 2, a number of design choices had to be made, as exploring all possible variations of these proposals may not be reasonably feasible. We still explored a few different possibilities. For instance, we experimented selecting hyper-parameters P, Q and R in equations (3.14) and (3.15) so that they can be different from each other in the multi-prototype approach (Section 3.3.2). We handled sentences by recasting zero-shot recognition as an image-sentence retrieval problem [156]. We tried phrase representation [102] in addition to single-word embeddings. We experimented with

other types of multi-modal representations [51]. However, results were either disappointing or did not provide a significant improvement over simpler methods, so they were not included.

Furthermore, the practical relevance of some of the proposed approaches can be debated. For instance, since we query the Flickr API to obtain the "visualness" associated with each word of a definition in the visualness approach (Section 3.3.1.1), one may legitimately argue that it may be easier to use this API to simply collect images from unseen classes, and thus get rid of the "zero-shot" hardship altogether. Nonetheless, we consider that our process still provides interesting insights and constitutes a challenging baseline. In addition, this method is not exclusive to Flickr and is transferable to other situations. Furthermore, the approach based on learned attention produces comparable results while being more applicable in practice.

Finally, despite the observed significant improvements, the performance of zero-shot recognition in the large scale scenarios explored in this chapter remains quite low. However, we consider that focusing on the semantic representation aspect of the zero-shot learning task is promising. We hope that in addition to the current focus on the zero-shot learning models themselves, future works will also take this aspect into consideration.

Conclusion

3.7 Summary of contributions

Throughout this manuscript, we focused on zero-shot learning as a means to decrease the need for human-provided annotations. We specifically focused on the more realistic generalized zero-shot learning setting, in which test classes can be both seen and unseen, as well as the more challenging large scale setting, in which semantic prototypes are obtained in an unsupervised manner.

In a first part, we identified several implicit assumptions frequently made by ranking methods, also called triplet loss methods, which may be detrimental to their performance. The first of these assumptions is the idea that different classes are equally distinct. In practice, some classes may be close to being indistinguishable even for a human, as illustrated on the CUB dataset. Penalizing confusions between such similar classes as much as confusions between very dissimilar classes may have adverse effects on the robustness of the learned multi-modal relations. We thus introduced a flexible semantic margin in the hinge rank loss, which depends on the distance between classes in the semantic space and takes attribute correlations into account. We further argued that in the usual formulation of the triplet loss, the actual value of the margin has little effect as the norm of the learned multi-modal projections can be arbitrarily scaled to compensate for the value of the margin. We thus proposed to constrain the norms of these projections, while still leaving some flexibility to the model with respect to the scale of these norms. Finally, zero-shot learning models usually consider that all training samples are relevant and should therefore be treated equally, while some instances may actually be highly atypical – again as illustrated on the CUB dataset. We introduced a simple weighting scheme in the training loss to take this effect into account. These different ideas led to the introduction of a reasonably simple zero-shot learning model based on a triplet loss. Notably, once trained, predictions can be made with either one or two elementary bi-linear projections from the visual features and semantic prototypes. State-of-the-art results were achieved in our experimental evaluation.

We then considered the gap between the accuracies on seen and unseen classes many models suffer from in a generalized zero-shot learning setting. We provided some theoretical and empirical insights on why this gap exists, and introduced a simple calibration process to reduce this gap and therefore increase performance. This process is based on a training-validation-testing split specific to generalized zero-shot learning and is applicable to most existing models. Experimental evaluations

showed a significant increase in performance for all evaluated models on all datasets. Importantly, this process can easily be combined with our proposed triplet loss model. This combination enables our model to equal or surpass the performance of generative methods, while being less restrictive in practice: while most best performing generative models require at least some additional training to assimilate new classes, our model can operate in a strictly class-inductive setting.

Some of the contributions in this first part were published in the proceedings of the International Conference on MultiMedia Modeling, 2019 [79], the others in the proceedings of the International Conference on Computer Vision, 2019 [80]. Our implementation of the triplet loss model has been released[9]. In addition, an adaptation of the survey of the state-of-the-art (Chapter 1) is in the process of being published as the chapter of a book on deep learning (release planned in early 2021).

In a second part, we focused on semantic representations in the context of large-scale zero-shot learning. In this context, class prototypes usually consist of word embeddings of the class names. We argued that embeddings trained on prevailing text corpora are likely to be unsuited for fine-grained zero-shot recognition. We thus proposed to collect new corpora with more visually oriented textual content by retrieving user-defined tags and image descriptions. We further proposed to adapt the usual skip-gram objective to take into account some phenomena related to this context, such as bulk-tagging. This adaptation can be reduced to some simple pre-processing steps, and thus enables to train most popular word embedding models with their original implementations. Experimental evaluation showed that the resulting embeddings lead to significantly better results in a large scale setting for most evaluated zero-shot learning models. However, evaluation on smaller scale datasets showed that performance is still considerably lower than performance obtained with hand-crafted attributes, particularly for the fine-grained dataset.

We then argued that using class name embeddings as semantic representations may have some hard limitations. We thus proposed to employ short descriptions in natural language as semantic information, and experimented with several approaches applicable to this task. In a first approach, we aimed to obtain a single prototype from a short sentence by using a weighted average of the constituting word embeddings. In a second approach, we instead adapted the triplet loss to enable a model to handle several semantic prototypes, and used the individual word embeddings as distinct

[9] `https://github.com/yannick-lc/iccv2019-triplet-loss`

prototypes. While qualitatively, the weights given to each word in a sentence seemed reasonable for both approaches, the quantitative results were disappointing. Indeed, both approaches led to results below the class name embedding baseline.

We finally experimented with the combination of prototypes obtained from class name embeddings and from short descriptions. We performed this with a simple convex combination. As recent large scale models make use of the class hierarchy, we further incorporated some information from parent classes. In spite of the poor performance of short definitions alone, this combination led to significantly better scores in our experimental evaluation.

Some contributions in this second part were published in the proceedings of the European Conference on Computer Vision Workshops, 2020 [81], and some are about to be published in the proceedings of the Asian Conference on Computer Vision, 2020 [82]. Corresponding semantic representations as well as the code used to generate them are in the process of being made publicly available on GitHub[10].

3.8 Perspectives

Even though they already enabled interesting results, many of our proposed contributions could be further explored. For instance, the current triplet loss model is based on elementary linear projections. Even though this simple architecture may be considered a strength, it could still be interesting to explore more complex, non linear architectures. Nonetheless, such architectures may require larger datasets to be efficiently trained. Additionally, as mentioned in Section 1.2.4, some generative methods [152, 162] incorporate a regression or classification loss besides the usual VAE or GAN losses. It may thus be worth it to try to combine high performing generative architectures with the proposed enhanced triplet loss.

Similarly, our proposed methods for employing short descriptions as semantic representations are fairly simple for now, even though they still lead to increased scores when combined with other types of embeddings. This "simplicity" is partly due to the limited amount of training data, as our models only had access to a thousand short sentences for training. With a larger training dataset, it is likely that other more complex approaches may produce better results. Since the combination of prototypes obtained from these simple methods with class name embeddings is already promising, it

[10] `https://github.com/yannick-lc/zsl-sentences`
`https://github.com/yannick-lc/semantic-embeddings-zsl`

seems reasonable to think that an approach leading to better results with sentences alone may produce exciting results when combined with class name embeddings.

Beyond these contributions to zero-shot learning, some ideas may also be transposed to different tasks. For instance, the tasks of person re-identification and multimedia retrieval also frequently make use of the hinge rank loss. It could therefore be beneficial to explore whether some contributions can be applied to these tasks. Applications to different multi-modal tasks in general such as Visual Question Answering (VQA), that similarly to ZSL aims at relating visual to textual information, may also be contemplated. On the other hand, advances in VQA may similarly have an impact on the field of ZSL. For instance, in the recent LXMERT architecture [146], a single model is trained on multiple tasks such as cross-modality matching and image question answering. Leveraging VQA datasets may thus enable to produce more robust visual-textual mappings from larger datasets, which may be exploited for zero-shot recognition.

Finally, we remain optimistic regarding the large-scale zero-shot learning scenario. Even though current scores are still arguably too low for many practical use cases, we hope our contributions will emulate further research regarding the semantic representations of classes. Indeed, this element seems to be a crucial point for large scale zero-shot learning scenarios, and yet very few works have focused on this aspect. Similarly to how zero-shot recognition benefited from advances in computer vision in general, it is also likely that advances in different fields such as natural language processing could benefit to this task. In the long term, an efficient large scale zero-shot recognition model based on unsupervised class prototypes would constitute a tremendous achievement in data frugality, and could greatly contribute to democratize the use of computer vision and machine learning.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for attribute-based classification. In *Computer Vision and Pattern Recognition*, pages 819–826, 2013.

[3] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, 2015.

[4] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *Computer Vision and Pattern Recognition*, 2015.

[5] Y. Annadani and S. Biswas. Preserving semantic relations for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7603–7612, 2018.

[6] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[7] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[8] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran. Zero-shot object detection. In *European Conference on Computer Vision*, 2018.

[9] R. H. Bartels and G. W. Stewart. Solution of the matrix equation ax+ xb= c [f4]. *Communications of the ACM*, 15(9):820–826, 1972.

[10] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[11] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In *Advances in neural information processing systems*, pages 899–907, 2013.

[12] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[13] J. Blitzer, D. P. Foster, and S. M. Kakade. Zero-shot domain adaptation: A multi-view approach. *Tech. Rep. TTI-TR-2009-1*, 2009.

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[15] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408, 2007.

[16] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[17] M. Bucher, S. Herbin, and F. Jurie. Generating visual representations for zero-shot classification. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2666–2673, 2017.

[18] M. Bucher, S. Herbin, and F. Jurie. Zero-shot classification by generating artificial visual features. In *RFIAP*, 2018.

[19] M. Bucher, V. Tuan-Hung, M. Cord, and P. Pérez. Zero-shot semantic segmentation. In *Advances in Neural Information Processing Systems*, pages 468–479, 2019.

[20] J. Callan. The lemur project and its clueweb12 dataset. In *Invited talk at the SIGIR 2012 Workshop on Open-Source Information Retrieval*, 2012.

[21] J. Callan, M. Hoy, C. Yoo, and L. Zhao. The clueweb09 dataset, 2009. *URL http://boston. lti. cs. cmu. edu/Data/clueweb09*, 2009.

[22] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5327–5336, 2016.

[23] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Classifier and exemplar synthesis for zero-shot learning. *International Journal of Computer Vision*, 128(1):166–201, 2020.

[24] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European Conference on Computer Vision*, pages 52–68. Springer, 2016.

[25] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference, BMVC 2014, Nottingham, UK, September 1-5, 2014*, 2014.

[26] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[27] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[28] L. Chen, H. Zhang, J. Xiao, W. Liu, and S.-F. Chang. Zero-shot visual recognition using semantics-preserving adversarial embedding networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1043–1052, 2018.

[29] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[30] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001.

[31] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[32] B. Demirel, R. G. Cinbis, and N. Ikizler-Cinbis. Zero-shot object detection by hybrid region embedding. In *British Machine Vision Conference (BMVC)*, 2018.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[34] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[35] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2584–2591, 2013.

[36] M. Elhoseiny, Y. Zhu, H. Zhang, and A. Elgammal. Link the head to the" beak": Zero shot learning from noisy text description at part precision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6288–6297. IEEE, 2017.

[37] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Computer Vision and Pattern Recognition*, pages 1778–1785. IEEE, 2009.

[38] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[39] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[40] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, 2013.

[41] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. Learning multimodal latent attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(2):303–316, 2013.

[42] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. Transductive multi-view zero-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2332–2345, 2015.

[43] Y. Fu, T. Xiang, Y.-G. Jiang, X. Xue, L. Sigal, and S. Gong. Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content. *IEEE Signal Processing Magazine*, 35(1):112–125, 2018.

[44] Y. Fu, Y. Yang, T. Hospedales, T. Xiang, and S. Gong. Transductive multi-label zero-shot learning. *arXiv preprint arXiv:1503.07790*, 2015.

[45] E. Gabrilovich, S. Markovitch, et al. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611, 2007.

[46] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[48] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[49] Y. Guo, G. Ding, J. Han, and Y. Gao. Synthesizing samples fro zero-shot learning. In *IJCAI*. IJCAI, 2017.

[50] Y. Guo, G. Ding, J. Han, and S. Tang. Zero-shot learning with attribute selection. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[51] T. Gupta, A. Schwing, and D. Hoiem. Vico: Word embeddings from visual co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7425–7434, 2019.

[52] A. Habibian, T. Mensink, and C. G. Snoek. Composite concept discovery for zero-shot video event detection. In *Proceedings of International Conference on Multimedia Retrieval*, pages 17–24, 2014.

[53] T. Hascoet, Y. Ariki, and T. Takiguchi. On zero-shot recognition of generic objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9553–9561, 2019.

[54] T. Hascoet, Y. Ariki, and T. Takiguchi. Semantic embeddings of generic objects for zero-shot learning. *EURASIP Journal on Image and Video Processing*, 2019(1):13, 2019.

[55] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[56] G. Hinton. Neural networks for machine learning. lecture 6a: Overview of mini-batch gradient descent. *Coursera lecture slides*, 2012.

[57] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, pages 857–864, 2003.

[58] Z. Hu, J. Tang, Z. Wang, K. Zhang, L. Zhang, and Q. Sun. Deep learning for image-based cancer detection and diagnosis- a survey. *Pattern Recognition*, 83:134–149, 2018.

[59] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *Computer Vision and Pattern Recognition*, pages 1629–1636, 2014.

[60] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing. Rethinking knowledge graph propagation for zero-shot learning. In *Computer Vision and Pattern Recognition*, pages 11487–11496, 2019.

[61] N. Karessli, Z. Akata, B. Schiele, and A. Bulling. Gaze embeddings for zero-shot image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4525–4534, 2017.

[62] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[63] A. Karpathy, A. Joulin, and L. F. Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in Neural Information Processing Systems*, pages 1889–1897, 2014.

[64] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[67] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[68] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[69] E. Kodirov, T. Xiang, Z. Fu, and S. Gong. Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2452–2460, 2015.

[70] E. Kodirov, T. Xiang, and S. Gong. Semantic autoencoder for zero-shot learning. In *Computer Vision and Pattern Recognition*, pages 4447–4456. IEEE, 2017.

[71] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1649–1652, 2009.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[73] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[74] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[75] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.

[76] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[77] H. Larochelle, D. Erhan, and Y. Bengio. Zero-data learning of new tasks. In *AAAI*, volume 2, pages 646–651, 2008.

[78] A. Lazaridou, G. Dinu, and M. Baroni. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280, 2015.

[79] Y. Le Cacheux, H. Le Borgne, and M. Crucianu. From classical to generalized zero-shot learning: A simple adaptation process. In *International Conference on Multimedia Modeling*, pages 465–477. Springer, 2019.

[80] Y. Le Cacheux, H. Le Borgne, and M. Crucianu. Modeling inter and intra-class relations in the triplet loss for zero-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10333–10342, 2019.

[81] Y. Le Cacheux, H. Le Borgne, and M. Crucianu. Using sentences as semantic representations in large scale zero-shot learning. *European Conference on Computer Vision Workshops*, 2020.

[82] Y. Le Cacheux, A. Popescu, and H. Le Borgne. Webly supervised semantic embeddings for large scale zero-shot learning. *Asian Conference on Computer Vision*, 2020.

[83] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[84] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[85] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[86] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[87] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88:365–411, 2004.

[88] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1576–1585, 2018.

[89] J. Lei Ba, K. Swersky, S. Fidler, et al. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4247–4255, 2015.

[90] X. Li and Y. Guo. Max-margin zero-shot learning for multi-class classification. In *Artificial Intelligence and Statistics*, pages 626–634, 2015.

[91] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[92] Y. Li, J. Zhang, J. Zhang, and K. Huang. Discriminative learning of latent features for zero-shot recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7463–7471, 2018.

[93] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR 2011*, pages 3337–3344. IEEE, 2011.

[94] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[95] Y. Lu. Unsupervised learning on neural network outputs: with application in zero-shot learning. *arXiv preprint arXiv:1506.00990*, 2015.

[96] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[97] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[98] T. Mensink, E. Gavves, and C. G. Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2441–2448, 2014.

[99] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.

[100] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[101] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

[102] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[103] G. A. Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[104] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[105] A. Mishra, S. Krishna Reddy, A. Mittal, and H. A. Murthy. A generative model for zero shot learning using conditional variational autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2188–2196, 2018.

[106] P. Nevavuori, N. Narra, and T. Lipping. Crop yield prediction with deep convolutional neural networks. *Computers and electronics in agriculture*, 163:104859, 2019.

[107] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[108] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. In *International Conference on Learning Representations*, pages 488–501, 2014.

[109] G. Obozinski, B. Taskar, and M. Jordan. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*, 2(2.2):2, 2006.

[110] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[111] N. O'Hare and V. Murdock. Modeling locations with social media. *Information retrieval*, 16(1):30–62, 2013.

[112] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.

[113] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[114] G. Patterson and J. Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2751–2758. IEEE, 2012.

[115] A. Paul, N. C. Krishnan, and P. Munjal. Semantically aligned bias reducing zero shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7056–7065, 2019.

[116] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[117] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[118] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

[119] J. Platt. Probabilities for sv machines, advances in large margin classifiers, 1999.

[120] A. Popescu, G. Etienne, and H. Le Borgne. Scalable domain adaptation of convolutional neural networks. preprint arXiv:1512.02013, 2015.

[121] A. Popescu and G. Grefenstette. Social media driven image retrieval. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, pages 1–8, 2011.

[122] R. Qiao, L. Liu, C. Shen, and A. Van Den Hengel. Less is more: zero-shot learning from online textual documents with noise suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2249–2257, 2016.

[123] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531, 2010.

[124] S. Rahman, S. Khan, and F. Porikli. Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts. In S. LNCS, editor, *Asian Conference on Computer Vision (ACCV)*, Perth, Australia, 2018.

[125] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society.

[126] S. Reed, Z. Akata, H. Lee, and B. Schiele. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016.

[127] R. Richardson, J. M. Schultz, and K. Crawford. Dirty data, bad predictions: How civil rights violations impact police data, predictive policing systems, and justice. *NYUL Rev. Online*, 94:15, 2019.

[128] M. Rohrbach, S. Ebert, and B. Schiele. Transfer learning in a transductive setting. In *Advances in neural information processing systems*, pages 46–54, 2013.

[129] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR 2011*, pages 1641–1648. IEEE, 2011.

[130] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015.

[131] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[132] 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. *International journal of computer vision*, 115(3):211–252, 2015.

[133] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[134] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[135] Y. Shigeto, I. Suzuki, K. Hara, M. Shimbo, and Y. Matsumoto. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer, 2015.

[136] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[137] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[138] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943, 2013.

[139] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[140] J. Song, C. Shen, Y. Yang, Y. Liu, and M. Song. Transductive unbiased embedding for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1024–1033, 2018.

[141] S. Srivastava, I. Labutov, and T. Mitchell. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316, 2018.

[142] G. Sumbul, R. G. Cinbis, and S. Aksoy. Fine-grained object recognition and zero-shot learning in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2):770–779, 2017.

[143] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[144] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[145] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[146] H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[147] The Guardian. Saola sighting in vietnam raises hopes for rare mammal's recovery. *The Guardian*, November 1999.

[148] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104, 2004.

[149] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484, 2005.

[150] N. Usunier, D. Buffoni, and P. Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th annual international conference on machine learning*, pages 1057–1064, 2009.

[151] W. N. van Wieringen. Lecture notes on ridge regression. *arXiv preprint arXiv:1509.09169*, 2015.

[152] V. K. Verma, G. Arora, A. Mishra, and P. Rai. Generalized zero-shot learning via synthesized examples. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4281–4289, 2018.

[153] V. K. Verma and P. Rai. A simple exponential family framework for zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 792–808. Springer, 2017.

[154] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 dataset, 2011.

[155] D. Wang, Y. Li, Y. Lin, and Y. Zhuang. Relational knowledge transfer for zero-shot learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[156] L. Wang, Y. Li, and S. Lazebnik. Learning deep structure-preserving image-text embeddings. In *Computer Vision and Pattern Recognition*, pages 5005–5013, 2016.

[157] W. Wang, V. W. Zheng, H. Yu, and C. Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019.

[158] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Computer Vision and Pattern Recognition*, pages 6857–6866, 2018.

[159] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

[160] J. Weston, C. Watkins, et al. Support vector machines for multi-class pattern recognition. In *European Symposium on Artificial Neural Networks*, volume 99, pages 219–224, 1999.

[161] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

[162] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5542–5551, 2018.

[163] Y. Xian, B. Schiele, and Z. Akata. Zero-shot learning-the good, the bad and the ugly. In *Computer Vision and Pattern Recognition*, pages 4582–4591, 2017.

[164] Y. Xian, S. Sharma, B. Schiele, and Z. Akata. f-vaegan-d2: A feature generating framework for any-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10275–10284, 2019.

[165] X. Xu, F. Shen, Y. Yang, D. Zhang, H. Tao Shen, and J. Song. Matrix tri-factorization with manifold regularizations for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3798–3807, 2017.

[166] E. Zablocki, P. Bordes, L. Soulier, B. Piwowarski, and P. Gallinari. Context-aware zero-shot learning for object recognition. In *International Conference on Machine Learning*, pages 7292–7303, 2019.

[167] H. Zhang and P. Koniusz. Zero-shot kernel learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7670–7679, 2018.

[168] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. In *Computer Vision and Pattern Recognition*, pages 2021–2030, 2017.

[169] Z. Zhang and V. Saligrama. Zero-shot recognition via structured prediction. In *European conference on computer vision*, pages 533–548. Springer, 2016.

[170] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1004–1013, 2018.

Appendix A

Additional details

A.1 Zero-shot learning datasets

We briefly present the zero-shot learning datasets we use throughout the manuscript.

Animals with Attributes or **AwA** [76] is one of the first proposed benchmarks for ZSL [75]. It contains 50 classes representing 50 animal species such as *antelope*, *grizzly bear* or *dolphin*. Class prototypes have 85 attributes such as *brown*, *stripes*, *hairless* or *claws*. Both binary and continuous attributes are provided with the dataset. The original dataset has recently been replaced by the very similar **AwA2** [161] due to copyright issues on some images. The latter contains the same animal species and attributes, for a total of 37322 images. 10 classes are used as unseen test classes, and the rest represents seen training classes and validation classes.

Importantly, it has been pointed out that 6 of the initial 10 unseen test classes from AwA are also among the 1000 classes from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [132]. Since classes from ILSVRC are frequently used to train deep convolutional neural network employed as visual feature extractors (Section 1.3.1), this induces an important bias in the case of zero-shot recognition, as these classes cannot be considered as truly unseen [163]. As a result, Xian *et al.* [163] introduced a new *"proposed split"* for this dataset such that no unseen test class is among the 1000 classes from ILSVRC. Similarly to most recent works [5, 164, 23], these are the splits we use in this manuscript.

Caltech UCSD Birds 200-2011 or **CUB** [154] is referred to as a "fine-grained" dataset, as its 200 classes all correspond to bird species (*black footed albatross*, *rusty blackbird*, *eastern towhee*...), and

many categories can be considered to be fairly similar. It contains a total of 11788 images. The class prototypes consist of 312 usually continuous attributes with values between 0 and 1. These continuous attributes are the class average of binary attributes provided for each image. Examples of attributes include *"has crown color blue"*, *"has nape color white"* or *"has bill shape cone"*. 50 classes are used as unseen test classes. Similarly to AwA2, we also employ the seen-unseen split proposed by Xian *et al.* [163].

SUN [114] is another example of a fine-grained dataset. It contains 717 classes representing different scenes such as *abbey*, *classroom*, *hospital* or *playground*, for a total of 14340 images. Class prototypes consist of 102 attributes, initially provided for each individual image and then averaged to obtain class representations. Examples of attributes include *fire*, cluttered space or diving. 72 classes are used as unseen test classes. Similarly to AwA2 and CUB, we employ the seen-unseen split proposed by Xian *et al.* [163].

The **ImageNet** [33] dataset has also been used as a large-scale ZSL benchmark [129, 40, 53]. This dataset contains classes as diverse as *coyote*, *goldfish*, *lipstick* or *speedboat*. Contrary to AwA or CUB, the usual semantic prototypes do not consist of manually-defined attributes. Instead, word embeddings of the class names are used as class representations – more details are provided in Section 1.3.2.

The seen training classes usually consist of the 1000 classes of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [132]. In the past, the approximately 20,000 remaining classes were used as unseen test classes. However, Hascoet *et al.* [53] recently showed that this induces a bias, in part due to the fact that unseen classes are often subcategories or supercategories of seen classes. The authors suggested instead to use only a subset of 500 of the total unseen classes such that they do not exhibit this problem. We use these 500 classes as unseen test classes in this manuscript.

Since classes from ImageNet correspond to WordNet "synsets" [103], additional information can be provided by the WordNet database. For instance, each synset is associated with a short definition. We make use of these definitions in Section 3.3. In addition, a word hierarchy is provided, with words representing hypernyms or hyponyms of other words. Some models exploit this information in the form of graph relations, as detailed in Section 1.3.2.

In addition to the four described above, other datasets have been employed as benchmarks for zero-shot recognition. For instance, **Attribute Pascal and Yahoo** or **aPY** [37] contains 32 generic classes

Parameter	Word2vec	GloVe	FastText
Epochs	25	100	25
Learning rate	0.1	0.05	0.1
Window	10	10	10
Embedding dimension	300	300	300

Table A.1 – Training parameters for the different semantic embedding models.

Model	**Command**
word2vec	-size 300 -window 1 -sample 1e-4 -negative 5 -hs 0 -binary 0 -cbow 0 -iter 25 -min-count 5
GloVe	-x-max 100 -iter 100 -eta 0.05 -vector-size 300 -alpha 0.75
FastText	skipgram -dim 300 -epoch 25 -minn 4 -maxn 6 -lr 0.1 -ws 10 -minCount 5

Table A.2 – Command lines used to train the embeddings.

such as *person*, *building* or *wolf* with 64 attributes. The **Oxford Flowers-102** dataset of **Flowers** [107] has been used for fine-grained recognition. Its semantic representations consist in 10 sentences per image as in [126], or in text from the corresponding Wikipedia pages as in [89]. The **Stanford Dogs** or **Dogs** [64] dataset has also been used as a fine-grained benchmark associated with corresponding Wikipedia pages [4]. However, these datasets are either less common, correspond to different settings with different semantic representations, or suffer from biases. For instance, most classes from aPY are in the 1000 classes from ILSVRC, which induces an even greater bias than in AwA as detailed above [163]. As a result, we do not make use of these datasets in this manuscript.

A.2 Implementation details

We provide additional implementation details regarding the parameters associated with the training of the word embedding models from Section 3.2.1 in tables A.1 and A.2.

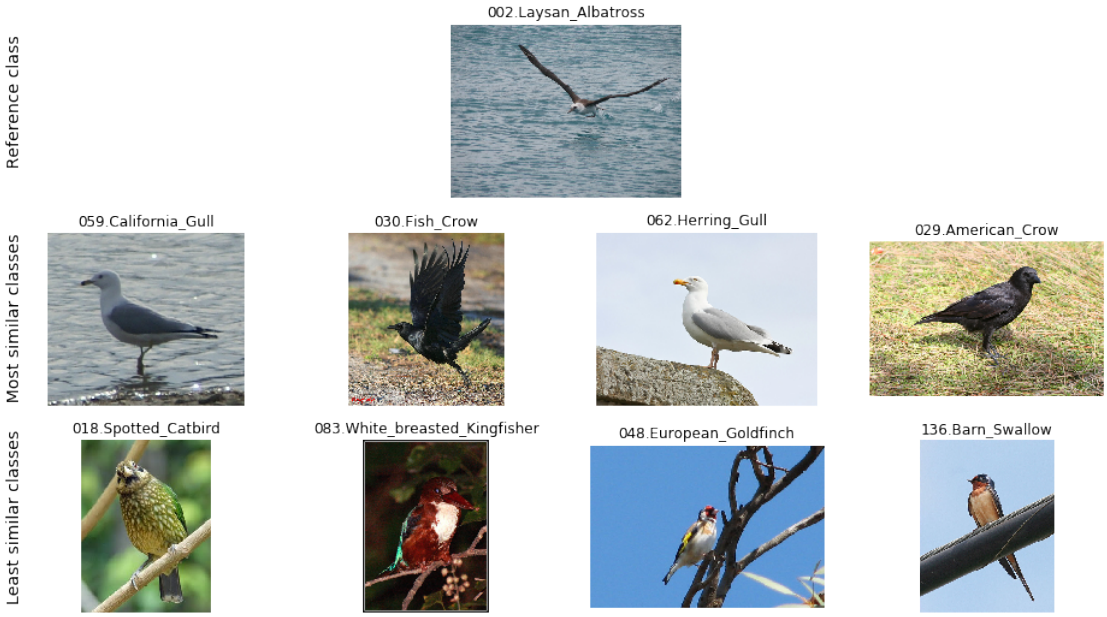

Figure A.1 – Top 4 most *(middle)* and least *(bottom)* similar classes to class *Laysan Albatros (top)*.

A.3 Illustrations

A.3.1 Illustrative examples for the semantic margin

We provide illustrative examples of the closest and farthest classes from a few additional classes from the CUB dataset, namely *Laysan Albatross*, *Least Auklet* (both present in Figure 2.1) and *Vesper Sparrow* in figures A.1, A.2 and A.3.

The distances are computed using the process described in Section 2.1. On average, closest and farthest classes seem to be reasonably consistent with what one would intuitively expect. Interestingly, classes *Fish Crow* and *American Crow* appear very often among the most similar classes, even for very different reference classes.

A.3.2 Illustrative examples for the relevance weighting

We also provide most and least relevant samples as measured by the relevance weighting scheme described in Section 2.3 for a few additional classes from CUB, again classes *Laysan Albatross*, *Least Auklet* and *Vesper Sparrow* in figures A.4, A.5 and A.6.

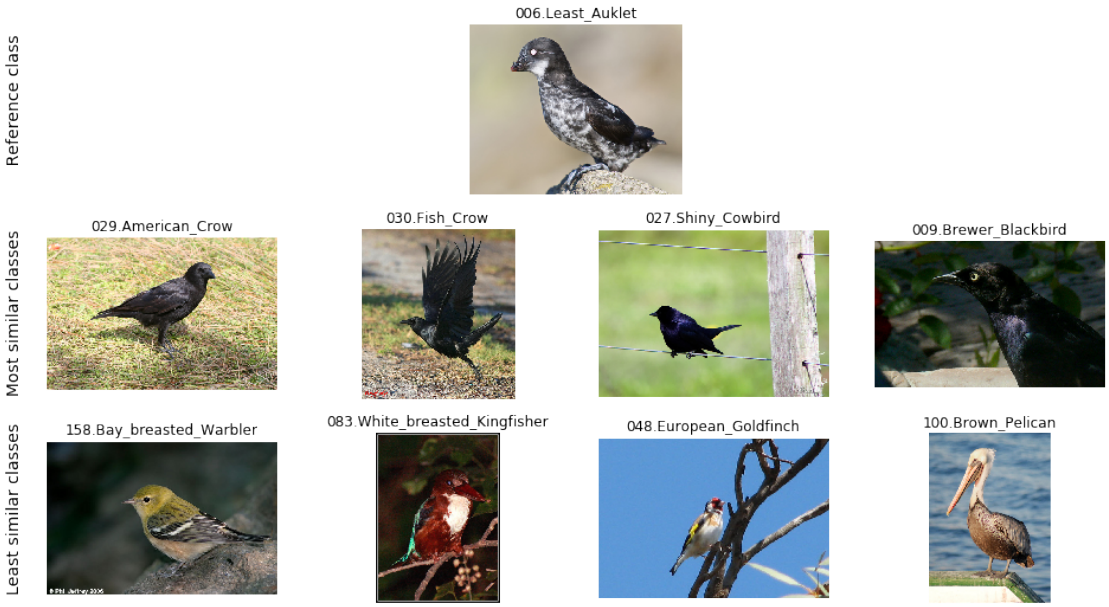

Figure A.2 – Top 4 most *(middle)* and least *(bottom)* similar classes to class *Least Auklet (top)*.

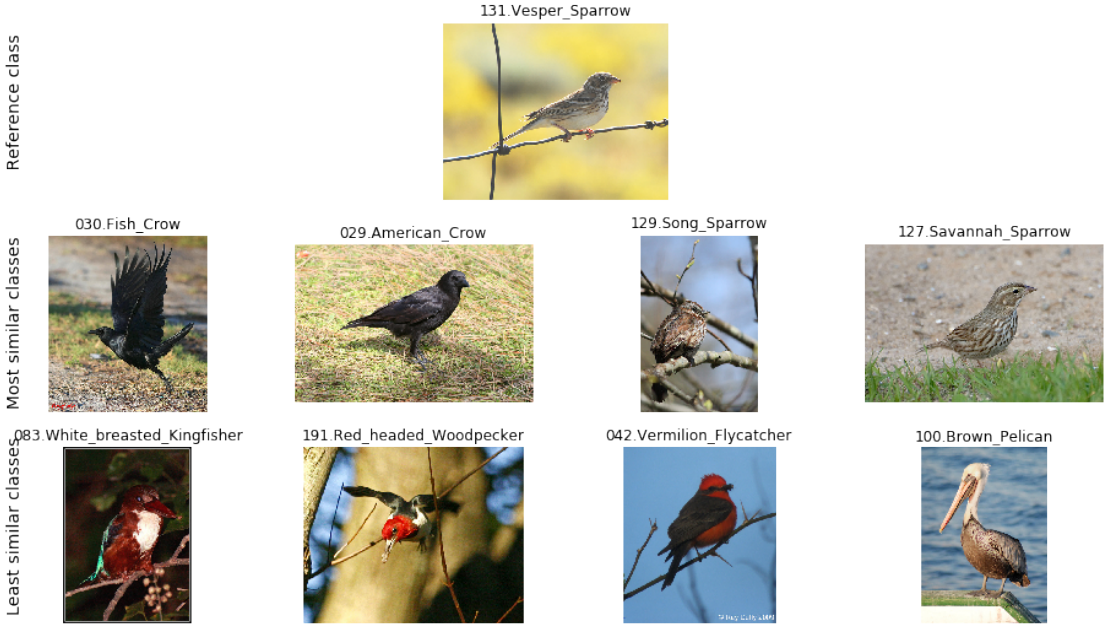

Figure A.3 – Top 4 most *(middle)* and least *(bottom)* similar classes to class *Vesper Sparrow (top)*.

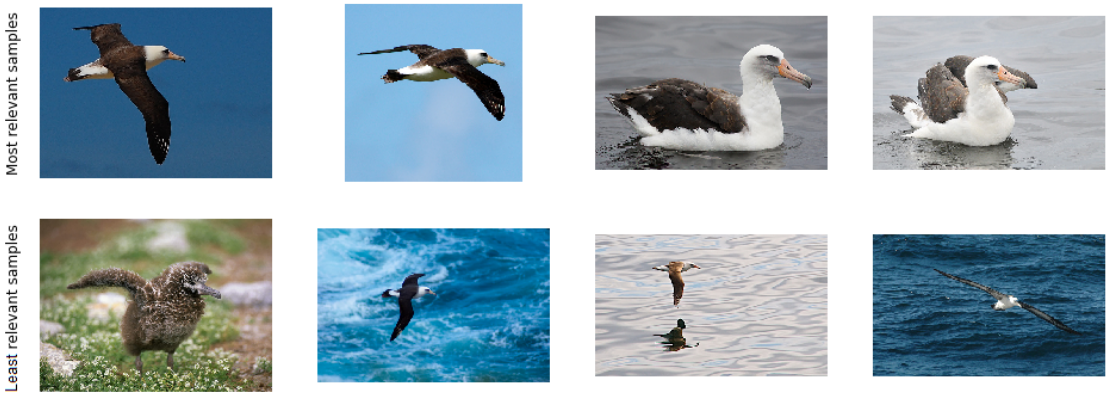

Figure A.4 – Top 4 most *(top)* and least *(bottom)* relevant samples for class *Laysan Albatros*

Figure A.5 – Top 4 most *(top)* and least *(bottom)* relevant samples for class *Least Auklet*

Most and least relevant samples for each class seem to be reasonably consistent. In particular, the nestling from Figure 2.1 is considered to be an outlier (Figure A.4). Other images considered as irrelevant include images of low quality (low resolution, over-saturated...), with atypical background or taken from unusual angles, e.g. focused only on a specific part of the bird.

It should be noted that if all images are relevant for a given class, some relevant images will be included in the "least relevant" examples. However, we consider that having a relevant image considered as irrelevant is not as detrimental as having an irrelevant image considered as relevant.

Figure A.6 – Top 4 most *(top)* and least *(bottom)* relevant samples for class *Vesper Sparrow*

A.3.3 ImageNet hierarchy

Figure A.7 represents the visualization of the full WordNet hierarchy for all 1000 (resp. 500) training (resp. testing) classes, as well as some intermediate nodes. Parts of this hierarchy are visible in Figure 3.4. We only keep one parent per node. For nodes which originally have several hypernyms, we keep the nodes corresponding to the longest path to the root node *"entity"*; we found that this leads to more meaningful paths, with fewer classes at each level. For example, we keep the path *"greyhound"* → *"hound"* → *"hunting_dog"* → *"dog"* → ... → *"animal"* (visible in Fig. A.7) instead of *"greyhound"* → *"racer"* → *"animal"*. We remove intermediate nodes which are not direct hypernyms of either a training or a testing class, as well as some other hand-picked nodes to improve readability.

It is interesting to observe that ZSL training and testing classes are not homogeneous in the hierarchy: some tree branches contain very few unseen classes, *e.g. "carnivore"*, while other contain many unseen classes and not a single seen class, *e.g. "woody_plant"*. These latter classes appear very challenging to correctly predict.

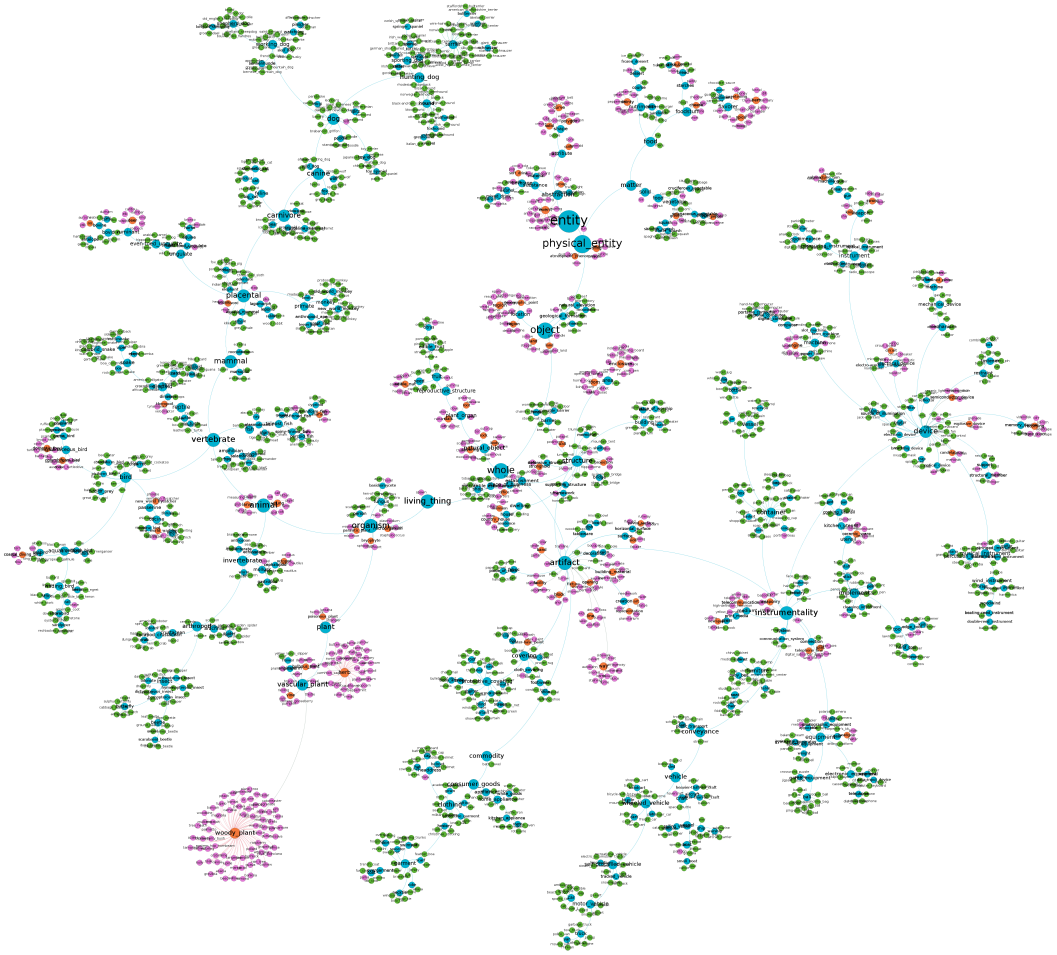

Figure A.7 – Overview of the full class hierarchy. Pink nodes refer to test classes, green nodes refer to train classes, orange nodes have only test classes below them and blue nodes are other intermediate nodes. Best viewed in color with at least 600% zoom.

Appendix B

Résumé en français

B.1 Introduction

Ces dernières années, la vision par ordinateur s'est imposée comme incontournable dans de nombreux domaines scientifiques comme industriels : les chèques bancaires sont généralement lus et traités automatiquement depuis des années [84, 85], les rendements agricoles peuvent être surveillés et anticipés via des images aériennes et satellite [106], la détection anticipée de tumeurs cancéreuses sur des images médicales pourrait bientôt contribuer à sauver des milliers de vies [58], la conduite autonome promet de révolutionner le secteur du transport [27], et de nombreuses autres innovations majeures ne demandent qu'à voir le jour.

Pratiquement toutes ces applications reposent sur des *réseaux neuronaux convolutifs profonds* [86]. Ces architectures d'apprentissage automatique ou *machine learning* produisent des caractéristiques (*features*) d'un niveau d'abstraction croissant, calculées séquentiellement à l'aide de noyaux de convolution dont les paramètres sont appris automatiquement par un modèle sur un grand nombre d'exemples d'entraînement. Malgré des performances inégalées, le besoin de données des modèles d'apprentissage profond est à l'origine de nouveaux défis. Le plus évident est l'important effort d'annotation requis pour fournir la grande quantité « d'étiquettes » (*labels*) nécessaires pour entraîner un modèle neuronal dans un contexte d'apprentissage supervisé. Cette contrainte peut être un frein à l'utilisation de ces architectures lorsque les ressources ou les moyens nécessaires pour mettre en œuvre un investissement aussi important ne sont pas disponibles. D'importants efforts de recherche ont été consacrés à ce problème. Par exemple, la tâche d'apprentissage « en quelques vues » (*few-shot learning*) vise à concevoir des modèles capables de reconnaître de nouvelles catégories après n'avoir eu accès qu'à quelques

exemples d'entraînement, généralement en réutilisant fortement des capacités précédemment acquises sur des problèmes source [99, 137]. L'apprentissage « une vue » (*one-shot learning*) est l'application extrême de cette idée, où seul un unique exemple d'apprentissage peut être utilisé par le modèle pour intégrer de nouvelles catégories [39, 74].

Et pourtant, la tâche d'apprentissage « zéro vue » ou *zero-shot learning* (ZSL) vise à pousser cette stratégie encore plus loin. Le but de cet exercice de frugalité ultime en termes de données est de concevoir des modèles capables de reconnaître des catégories visuelles pour lesquelles *aucun* exemple d'apprentissage n'est fourni [77, 75, 112]. Le principe de base peut être illustré par la capacité humaine à relier des contenus visuels et non visuels. À titre d'exemple, une personne n'ayant jamais vu une seule image ou une seule représentation d'un tigre – et n'en ayant naturellement jamais rencontré physiquement – devrait être en mesure d'en reconnaître un instantanément si on lui fournit l'information selon laquelle un tigre est similaire à un (très) gros chat orange avec des rayures noires et un ventre blanc. De toute évidence, une information sémantique similaire à la description du « chat orange rayé » concernant la classe *tigre* est nécessaire pour que la reconnaissance « zéro-shot » soit possible. En ce sens, les principes de base de l'apprentissage zéro-shot sont en fait assez différents de l'apprentissage « few-shot » et « one-shot », car ces tâches sont généralement purement visuelles. À l'inverse, l'apprentissage zéro-shot est par essence une tâche *multimodale*, qui nécessite la capacité de faire le lien entre une modalité visuelle (c'est-à-dire des images) et au moins une modalité non visuelle (par exemple du texte, des attributs...). Plus précisément, dans ce document, nous considérons que le terme *zero-shot learning* fait référence à la conception et à l'entraînement d'un modèle dont le but est de classifier des images appartenant à des classes *non vues* (*unseen*), pour lesquelles aucun exemple d'entraînement n'est fourni. Pour ce faire, pendant la phase d'entraînement, le modèle dispose d'exemples appartenant aux classes *vues*, qui sont strictement différentes des classes non vues. Des informations sémantiques sont en outre fournies pour les classes vues et non vues.

Historiquement, ce concept d'apprentissage zéro-shot est apparu il y a plus de dix ans avec les travaux de pionniers tels que Larochelle *et al.* [77], qui ont montré qu'il était possible de reconnaître automatiquement des classes de test distinctes des classes d'entraînement, ainsi que Lampert *et al.* [75], qui ont utilisé des attributs tels que *« noir »*, *« orange »* ou *« à rayures »* pour reconnaître des images d'espèces animales pour lesquelles aucun exemple d'apprentissage n'était fourni au modèle.

Ce nouveau défi a rapidement suscité l'intérêt de la communauté de vision par ordinateur, et de nouveaux modèles et jeux de données d'évaluation ont été rapidement proposés [129, 99, 2, 40, 138, 108]. Différentes tâches et contextes ont été pris en compte: Socher *et al.* [138] ont introduit un mécanisme de détection de nouveauté, afin que les modèles puissent reconnaître à la fois les classes non vues *et* vues, un contexte désormais connu sous le nom d'apprentissage zéro-shot *généralisé* (*generalized zero-shot learning* ou GZSL[1]) [24]. En tant que méthode visant à réduire considérablement la quantité de données nécessaires pour entraîner des modèles, l'apprentissage zéro-shot est d'autant plus pertinent dans un contexte « large échelle ». Par conséquent, Rohrbach *et al.* [129] ont proposé d'employer 200 des 1000 classes du challenge ILSVRC (*ImageNet large scale visual recognition challenge* [132]) comme classes de test non vues, en utilisant les informations hiérarchiques de WordNet pour créer des représentations de classe. Frome *et al.* [40] sont allés encore plus loin, en utilisant les 1000 classes d'ILSVRC comme des classes d'entraînement et les 20 000 classes supplémentaires d'ImageNet comme classes de test non vues. Comme il est généralement difficile de fournir des attributs pour des milliers de classes, des représentations sémantiques adaptées sont nécessaires dans un contexte de si grande échelle. Ces représentations prennent généralement la forme de plongements lexicaux (*word embeddings* [100, 102]), des représentations lexicales vectorielles conservant des propriétés sémantiques intéressantes. Ces plongements ont l'avantage notoire de provenir de modèles entraînés sur des corpus de texte conséquents de manière non supervisée, et donc de ne nécessiter pratiquement aucun effort humain d'annotation. Il est donc désormais standard d'appliquer ces modèles d'embeddings pré-entraînés aux noms des classes pour obtenir des représentations sémantiques dans un contexte d'apprentissage zéro-shot à grande échelle [40, 138, 108].

La capacité de reconnaître efficacement des images dans ce contexte pourrait sans doute être considérée comme le « Graal » des approches économes en efforts humains, car on pourrait ainsi envisager de produire des modèles capables de reconnaître des milliers de classes ne nécessitant quasiment aucun effort d'annotation. Cependant, en pratique, les performances de ces modèles restent modestes, et les performances rapportées sur les benchmarks grande échelle standard sont sans doute trop faibles pour de nombreux cas d'applications pratiques [53]. En général, les performances des modèles d'apprentissage zéro-shot sont sans surprise inférieures à celles des modèles supervisés standard [164]. En outre, la plupart des approches d'apprentissage zéro-shot ont tendance à souffrir de

[1]La liste des abréviations fréquemment utilisées dans ce document est disponible table 1.3

limitations supplémentaires. Par exemple, dans un contexte d'apprentissage zéro-shot généralisé plus réaliste dans lequel les classes de test peuvent être vues ou non vues, de nombreux modèles existants ont tendance à prédire les classes vues beaucoup plus fréquemment que les classes non vues [24, 163], ce qui diminue considérablement les performances sur ces dernières et donc l'intérêt d'utiliser la reconnaissance zéro-shot. Ce déséquilibre entre les classes vues et non vues est en partie réduit avec les approches génératives récentes [17, 152, 162], mais cela se fait au prix d'hypothèses plus restrictives, car contrairement à d'autres approches, l'ajout de nouvelles classes nécessite souvent un réentraînement au moins partiel pour ces modèles.

Dans cette thèse, nous tentons de surmonter certaines de ces limites à un apprentissage zéro-shot à grande échelle utilisable en pratique. Nous analysons les approches existantes de l'apprentissage zéro-shot, et en particulier les modèles basés sur des fonctions de coût dites de « triplet » ou de classement (*hinge rank loss*). Dans une première partie (section B.2 ou sections 2.1 à 2.5 du chapitre 2), nous défendons l'idée selon laquelle les modèles appartenant à cette famille font généralement plusieurs hypothèses implicites concernant la nature des classes et des exemples d'apprentissage, et que ces hypothèses peuvent ne pas être justifiées en pratique. Par exemple, ces modèles considèrent généralement que toutes les classes sont « pareillement différentes », ce qui signifie qu'aucune paire de classes distinctes n'est considérée comme plus similaire qu'une autre paire. Au contraire, nous soutenons que ce n'est souvent pas le cas en pratique et que ne pas tenir compte de cet aspect peut être préjudiciable à la performance du modèle.

Dans une seconde partie (section B.3 ou sections 2.6 à 2.7 du chapitre 2), nous nous intéressons à l'écart de performances entre les classes vues et non vues dans un cadre d'apprentissage zéro-shot généralisé. Nous proposons un processus simple pour réduire la différence de performance entre les instances des classes vues et non vues. L'approche proposée a également l'avantage de permettre d'ajouter sans effort de nouvelles classes non vues à un modèle déjà entraîné : contrairement à la plupart des approches génératives existantes, aucun apprentissage supplémentaire n'est requis.

En partant du constat qu'un des objectifs du ZSL est de minimiser les efforts d'annotation, nous nous intéressons dans une troisième partie (section B.4 ou sections 3.1 à 3.2 du chapitre 3) au rôle des représentations sémantiques obtenues de manière non supervisée typiquement employées à grande échelle. Cet aspect est étonnamment sous-étudié dans la littérature actuelle. Nous argumentons que

les corpus de texte génériques peuvent être inadaptés pour l'obtention d'embeddings intégrant les propriétés visuelles des mots, et proposons à la place de nouveaux corpus ainsi qu'une méthode de prétraitement appropriée.

Néanmoins, en dépit de résultats significativement améliorés par la méthode proposée, nous défendons que l'emploi d'embeddings de noms de classes en tant que représentations sémantiques peut se heurter à des limitations difficilement surmontables. Dans une quatrième et dernière partie (section B.5 ou sections 3.3 à 3.5 du chapitre 3), nous proposons ainsi un compromis entre l'emploi de plongements non supervisés ne nécessitant absolument aucun effort humain et la conception laborieuse d'attributs exhaustifs, sous la forme de l'utilisation de descriptions courtes en langage naturel. Nous proposons plusieurs approches pour exploiter de telles descriptions, et optons finalement pour des prototypes sémantiques constitués de combinaisons de représentations non supervisées et de descriptions courtes. Nous montrons que cette combinaison permet d'obtenir des résultats « à l'état de l'art » dans un cadre d'apprentissage zéro-shot à grande échelle, tout en maintenant la quantité requise d'effort humain d'annotation à un niveau relativement raisonnable.

Il est à noter que par nature, ce résumé en français contient moins de détails que le texte original complet en anglais. En particulier, certains détails d'implémentation non essentiels sont parfois omis dans un souci de concision. Nous référons les lecteurs intéressés par ces détails, par exemple dans le but de reproduire certains de nos résultats, au texte principal en anglais.

B.2 Hypothèses implicites dans les méthodes de classement

Tout au long de ce document, la notation $\mathbf{x}$ représentera un vecteur de caractéristiques visuelles d'une image, obtenu par exemple en utilisant un réseau de neurones convolutif profond tel que ResNet [55] comme extracteur de caractéristiques [125, 25]. La notation y désignera l'étiquette (*label*) associée à cette image, représentant la classe de l'image (par exemple *tigre*, *zèbre*...). Le prototype sémantique d'une classe c sera représenté par $\mathbf{s}_c$. Un tel prototype peut par exemple consister en un vecteur d'attributs tels que *« est orange »*, *« a des rayures »* et *« a des sabots »*, auquel cas la classe *tigre* pourrait être représentée par $(1\ 1\ 0)^\top$. Ces notations ainsi que d'autres sont résumées dans le tableau 1.2.

De nombreux modèles sont basés sur une fonction de compatibilité f entre la représentation d'une

image $\mathbf{x}$ et la représentation sémantique d'une classe $\mathbf{s}$. Par exemple, les méthodes de basées sur un « coût de triplet » ou coût de classement (*triplet loss* ou *hinge rank loss*) peuvent être présentées comme visant à répondre à l'intuition selon laquelle chaque représentation visuelle $\mathbf{x}$ devrait être « beaucoup plus » compatible avec le prototype $\mathbf{s}_c$ correspondant à sa classe c qu'avec tous les autres prototypes étant donné une fonction de compatibilité f. Plus formellement, ces méthodes ont pour objectif d'imposer la contrainte $f(\mathbf{x}, \mathbf{s}_y) \gg f(\mathbf{x}, \mathbf{s}_c)$, avec $(\mathbf{x}, y)$ une représentation visuelle étiquetée, $\mathbf{s}_y$ le prototype sémantique correspondant et $\mathbf{s}_c$ un autre prototype ($c \neq y$). Ceci est appliqué en visant à ce que $f(\mathbf{x}, \mathbf{s}_y) \geq m + f(\mathbf{x}, \mathbf{s}_c)$, où m est une marge fixée. Cet objectif peut être exprimé par la fonction de coût

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{s}_c, \mathbf{s}_y; f) = [m + f(\mathbf{x}, \mathbf{s}_c) - f(\mathbf{x}, \mathbf{s}_y)]_+ \tag{B.1}$$

Une implémentation simple d'un fonction de coût de triplet telle que DeViSE [40] consiste à utiliser une fonction de compatibilité bilinéaire $f_{\mathbf{W}}(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \mathbf{W}\mathbf{s}$ et à sommer le coût de l'équation (B.1) sur toutes les combinaisons d'exemples visuels et de prototypes de classe de l'ensemble d'entraînement:

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{\substack{c \in \mathcal{C}^{\mathcal{S}} \\ c \neq y_n}} [m + f(\mathbf{x}_n, \mathbf{s}_c) - f(\mathbf{x}_n, \mathbf{s}_{y_n})]_+ \tag{B.2}$$

Plusieurs variations de cette idée ont été proposées [4, 3]. Cependant, bien que ces méthodes aient conduit à des résultats prometteurs pour la tâche de ZSL, nous défendons dans cette section l'idée selon laquelle ces méthodes ne prennent pas en compte plusieurs aspects importants du problème en raison d'hypothèses implicites.

La première hypothèse de ce type est que *les classes sont pareillement distinctes*, car il n'y a aucune différence entre deux affectations de classe incorrectes dans le coût de triplet de l'équation (B.1). Cependant, dans de nombreux cas, et en particulier dans les jeux de données à granularité fine comprenant de nombreuses classes, il peut y avoir des groupes de classes très similaires. La figure 2.1 illustre un tel cas.

Pour améliorer la robustesse de la correspondance apprise entre les espaces sémantique et visuel, nous proposons de remplacer la marge fixe m dans l'équation (B.1) par une marge variable $m(c, c')$ mesurant la (dis)similarité entre les classes c et c'. De cette façon, pour des classes c et c' très similaires avec une dissemblance proche de 0, il suffit que $f(\mathbf{x}, \mathbf{s}_c) > f(\mathbf{x}, \mathbf{s}_{c'})$, c étant la classe associée

à $\mathbf{x}$. Inversement, des classes très différentes devraient être faciles à distinguer, et nous nous attendons à ce que $f(\mathbf{x}, \mathbf{s}_c) > M + f(\mathbf{x}, \mathbf{s}_{c'})$, avec $M = m(c, c')$ très grand.

Nous proposons de mesurer cette similarité dans l'espace sémantique. Comme les attributs ont tendance à être corrélés [59], nous utilisons une distance de Mahalanobis pour prendre en compte ces corrélations. La différence entre les classes i et j avec les prototypes respectifs $\mathbf{s}_i$ et $\mathbf{s}_j$ est donc

$$m(i, j) = \Big((\mathbf{s}_i - \mathbf{s}_j)\mathbf{\Sigma}^{-1}(\mathbf{s}_i - \mathbf{s}_j)\Big)^{\frac{1}{2}} \tag{B.3}$$

où $\mathbf{\Sigma}^{-1}$ est l'inverse de la matrice de covariance entre les attributs, qui peut être estimée en utilisant les prototypes des classes vues. Nous obtenons une estimation robuste de $\mathbf{\Sigma}^{-1}$ en utilisant la méthode de Ledoit-Wolf [87]. Nous proposons également de réajuster les distances pour avoir une moyenne $\mu_{\mathbf{M}}$ et un écart type $\sigma_{\mathbf{M}}$ (équation (2.7)), où $\mu_{\mathbf{M}}$ et $\sigma_{\mathbf{M}}$ sont considérés comme des hyperparamètres du modèle. Une illustration des distances d'origine (à gauche) et remises à l'échelle avec la moyenne $\mu_{\mathbf{M}}$ et l'écart-type $\sigma_{\mathbf{M}}$ définis (à droite) est présentée figure 2.2. Une illustration des classes les plus et les moins similaires aux classes *"red-legged kittiwake"* et *"arctic stern"* du jeu de données CUB [154] est disponible figure 2.3.

Dans l'équation (B.1), la marge m est censée agir comme un régulariseur et réduire le surajustement (*overfitting*) sur l'ensemble d'apprentissage. Intuitivement, on s'attend à ce qu'une plus grande valeur de la marge m incite le modèle à augmenter la différence $f(\mathbf{x}, \mathbf{s}_y) - f(\mathbf{x}, \mathbf{s}_c)$ entre la compatibilité de la paire correspondante $f(\mathbf{x}, \mathbf{s}_y)$ et la compatibilité de la paire non correspondante $f(\mathbf{x}, \mathbf{s}_c)$, et permette ainsi d'améliorer la robustesse du modèle – du moins jusqu'à ce que la contrainte imposée par l'équation (B.1) devienne non satisfiable. C'est ce que nous appelons *l'hypothèse que la marge est un régulariseur efficace*. Cependant, une plus grande valeur de m peut produire des effets différents non désirables, en particulier avec les fonctions de compatibilité de produit scalaire.

La fonction de compatibilité f prend souvent la forme d'un produit scalaire entre la projection $\boldsymbol{\theta}(\mathbf{x})$ d'une représentation visuelle $\mathbf{x}$ et la projection $\boldsymbol{\phi}(\mathbf{s})$ d'un prototype sémantique $\mathbf{s}$:

$$f(\mathbf{x}, \mathbf{s}) = \boldsymbol{\theta}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{s}) \tag{B.4}$$

Par exemple, pour les modèles avec une fonction de compatibilité linéaire tels que DeViSE, nous avons $\boldsymbol{\theta}(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ et $\boldsymbol{\phi}(\mathbf{s}) = \mathbf{s}$, ce qui donne la fonction de compatibilité bilinéaire $f(\mathbf{x}, \mathbf{s}) = \mathbf{x}^\top \mathbf{W}\mathbf{s}$. Ce

produit scalaire $\boldsymbol{\theta}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{s})$ peut également être écrit

$$\boldsymbol{\theta}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{s}) = \|\boldsymbol{\theta}(\mathbf{x})\|_2 \cdot \|\boldsymbol{\phi}(\mathbf{s})\|_2 \cdot \cos(\alpha) \tag{B.5}$$

où α est l'angle entre $\boldsymbol{\theta}(\mathbf{x})$ et $\boldsymbol{\phi}(\mathbf{s})$.

Ainsi, la valeur de la compatibilité $f(\mathbf{x}, \mathbf{s})$ dépend de trois composantes: la norme $\|\boldsymbol{\theta}(\mathbf{x})\|_2$ de la projection de la réprésentation visuelle, la norme $\|\boldsymbol{\phi}(\mathbf{s})\|_2$ de la projection du prototype sémantique, et le cosinus $\cos(\alpha)$ de l'angle entre les projections $\boldsymbol{\theta}(\mathbf{x})$ et $\boldsymbol{\phi}(\mathbf{s})$. Le cosinus de l'angle α est de toute évidence borné. Puisque $\mathbf{s}$ est généralement normalisé pour avoir une norme unitaire, il en est de même pour $\boldsymbol{\phi}(\mathbf{s})$ lorsque $\boldsymbol{\phi}$ est l'identité. Cependant, la norme de $\boldsymbol{\theta}(\mathbf{x})$ n'est pas bornée.

En pratique, cela signifie que si l'on double la marge m par rapport à un modèle de base, le nouveau modèle peut simplement doubler la différence $f(\mathbf{x}, \mathbf{s}_y) - f(\mathbf{x}, \mathbf{s}_c)$ dans l'équation (B.1) en doublant la compatibilité $f(\mathbf{x}, \mathbf{s})$ pour tout $\mathbf{s}$, ce qui peut être obtenu en doublant simplement la norme $\|\boldsymbol{\theta}(\mathbf{x})\|_2$ de la projection $\boldsymbol{\theta}(\mathbf{x})$. Cela peut aboutir à un nouveau modèle très similaire au modèle de base malgré une valeur différente de m, où les valeurs de $\boldsymbol{\theta}(\mathbf{x})$ ont simplement été doublées. Par conséquent, la valeur réelle de la marge m a peu d'impact sur les paramètres appris par le modèle.

Cet effet est visible empiriquement tel qu'illustré figure 2.4. Cela rend la valeur de la marge m peu pertinente, et réduit ainsi la régularisation fournie par la marge.

Pour résoudre ce problème, nous introduisons une fonction de *normalisation partielle* $\boldsymbol{\Psi}$, prenant un vecteur $\mathbf{v}$ comme entrée et paramétrée par un scalaire $\rho \in [0, 1]$ tel que

$$\boldsymbol{\Psi}_\rho(\mathbf{v}) = \frac{1}{\rho\,(\|\mathbf{v}\|_2 - 1) + 1} \cdot \mathbf{v} \tag{B.6}$$

Une valeur de $\rho = 0$ signifie qu'aucune transformation n'est appliquée à $\mathbf{v}$, et une valeur de $\rho = 1$ signifie que $\mathbf{v}$ est entièrement normalisé pour avoir une norme euclidienne unitaire. Les valeurs comprises entre 0 et 1 ont des résultats intermédiaires. Cette norme partielle est combinée avec une régularisation sur $\boldsymbol{\theta}(\mathbf{x})$ – par exemple une pénalité $\ell 1$ ou $\ell 2$.

La valeur de ρ est alors considérée comme un hyperparamètre du modèle, qui permet de contrôler dans quelle mesure nous permettons au modèle d'ajuster l'échelle les projections d'instances visuels pour accommoder une plus grande marge. La figure 2.4 montre l'effet de différentes valeurs de ρ sur l'ajustement de l'échelle de la norme de la projection $\boldsymbol{\theta}(\mathbf{x})$ par rapport à la marge m.

La troisième hypothèse faite par les modèles basés sur un coût de triplet, et plus largement par la plupart des modèles de ZSL, est que tous les exemples d'apprentissage sont pertinents. Cela signifie que tous les exemples des classes vues sont considérés comme pareillement représentatifs lors de l'apprentissage du modèle. Cependant, comme illustré figure 2.1, certains exemples peuvent être assez différents de la plupart des éléments de leur classe, ce qui peut avoir des effets indésirables pendant l'entraînement.

Nous proposons d'attribuer un score à chaque exemple d'apprentissage pour quantifier sa « représentativité » par rapport à sa classe, ainsi que de pondérer le coût associé à chaque exemple d'apprentissage dans la fonction objectif à l'aide de ce score. Pour chaque classe c, nous calculons la représentation visuelle moyenne $\overset{*}{\mathbf{x}}_c$. Pour chaque exemple $\mathbf{x}_m^c$ appartenant à cette classe, nous calculons ensuite la distance u_m^c à la représentation visuelle moyenne $\overset{*}{\mathbf{x}}_c$ de la classe c dans l'espace visuel.

A condition que les caractéristiques visuelles conviennent pour que ces distances aient un sens, cela fournit une première estimation du degré de différence entre une image et les autres images de la même classe. Nous normalisons ces distances pour qu'elles soient à peu près sur la même échelle quelle que soit la variance inter-classe et fixons les distances u_m^c de sorte qu'elles aient une moyenne nulle et une variance unitaire pour toutes les classes.

Nous définissons enfin les poids de représentativité v_m^c en fonction des distances redimensionnées u_m^c :

$$v_m^c = 1 - \Phi\left(u_m^c\right) \tag{B.7}$$

où $\Phi(\cdot)$ est la fonction de représentation de la distribution normale. De cette façon, les instances très éloignées du centre de gravité de leur classe ont un poids proche de 0, et les instances très proches du centre de gravité ont un poids proche de 1. Un exemple de distribution de tous les poids v_m^c est représenté figure 2.5. Une illustration des instances les plus pertinentes et les moins pertinentes pour les classes *"red-legged kittiwake"* et *"arctic stern"* telles que mesuré par les poids de représentativité est disponible figure 2.6.

Nous pouvons unifier ces idées dans un modèle de ZSL basé sur un coût de triplet. Le modèle consiste à apprendre les projections $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$ qui projettent respectivement des échantillons visuels $\mathbf{x}$ et des prototypes sémantiques $\mathbf{s}$ vers un espace commun, de sorte que la compatibilité entre $\mathbf{x}$ et $\mathbf{s}$ puisse être évaluée dans cet espace avec un produit scalaire. La normalisation partielle $\boldsymbol{\Psi}_\rho$ (equation (B.6))

est appliquée aux projections résultantes $\boldsymbol{\theta}(\mathbf{x})$ et $\boldsymbol{\phi}(\mathbf{s})$ avant d'effectuer le produit scalaire. On a donc une fonction de compatibilité f telle que

$$f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_n, \mathbf{s}_c) = \boldsymbol{\Psi}_\rho(\boldsymbol{\theta}(\mathbf{x}_n))^\top \boldsymbol{\Psi}_\rho(\boldsymbol{\phi}(\mathbf{s}_c)) \tag{B.8}$$

En utilisant $m(c, c')$ de l'équation (B.3) (et (2.7)) comme marge sémantique souple entre deux classes c et c', pour un triplet $(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c)$, $c \neq y_n$, le coût de triplet prend maintenant la forme

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c; f_{\boldsymbol{\theta},\boldsymbol{\phi}}) = [m(y_n, c) + f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_n, \mathbf{s}_c) - f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_n, \mathbf{s}_{y_n})]_+ \tag{B.9}$$

Nous adoptons l'approche la plus simple pour évaluer ce coût sur l'ensemble d'apprentissage, qui consiste à simplement additionner ce coût sur tous les triplets d'apprentissage $(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c)$, $c \neq y_n$, dans un cadre similaire à DeViSE. La coût de chaque triplet est en outre pondéré par le poids de représentativité v_n de l'équation (B.7). La fonction de coût finale qui en résulte est :

$$\frac{1}{N}\sum_{n=1}^{N}\left(v_n \sum_{\substack{c\in\mathcal{C}^\mathcal{S} \\ c\neq y_n}} \mathcal{L}_{\text{triplet}}(\mathbf{x}_n, \mathbf{s}_{y_n}, \mathbf{s}_c; f_{\boldsymbol{\theta},\boldsymbol{\phi}})\right) + \lambda\Omega[f_{\boldsymbol{\theta},\boldsymbol{\phi}}] \tag{B.10}$$

où λ est un hyperparamètre contrôlant le poids de la régularisation Ω. $\Omega[f_{\boldsymbol{\theta},\boldsymbol{\phi}}]$ est défini comme la somme des normes[2] $\ell 1$ des paramètres respectifs $\theta_1, \ldots, \theta_P$ et $\phi_1, \ldots, \phi_Q$ des projections $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$:

$$\Omega[f_{\boldsymbol{\theta},\boldsymbol{\phi}}] = \frac{1}{P}\sum_{p=1}^{P}|\theta_p| + \frac{1}{Q}\sum_{q=1}^{Q}|\phi_q| \tag{B.11}$$

Dans notre implémentation, nous utilisons de simples projections linéaires pour $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$. Nous considérons *deux variantes* du modèle : dans la première appelée $\boldsymbol{\theta} + \mathbf{I}$, nous projetons uniquement les caractéristiques visuelles $\mathbf{x}$ sur l'espace sémantique. Dans la deuxième variante $\boldsymbol{\theta} + \boldsymbol{\phi}$, nous projetons linéairement à la fois $\mathbf{x}$ et $\mathbf{s}$ sur un espace commun de même dimension K que l'espace sémantique. On peut considérer que la variante choisie est un hyperparamètre, nous sélectionnons donc la variante avec le meilleur score sur l'ensemble de validation.

Bien que ce cadre introduise un certain nombre d'hyperparamètres qui peuvent sembler difficiles à sélectionner, cette approche peut être vue comme étant simplement une généralisation de DeViSE ou d'autres modèles simples basés sur un coût de triplet. Un avantage de cette approche est que

[2]De manière similaire à Chao *et al.* [22], une régularisation $\ell 1$ peut introduire de la parcimonie de sorte que seul un sous-ensemble d'attributs est utilisé.

nous apprenons simplement une fonction de compatibilité (bi)linéaire. Une fois le modèle entraîné, des prédictions peuvent être effectuées aussi simplement que pour n'importe quel modèle basé sur une fonction de compatibilité directe : étant donné une instance de test $\mathbf{x}$, nous prédisons la classe $\hat{y}$ telle que

$$\hat{y} = \underset{c \in \mathcal{C}^U}{\operatorname{argmax}} \; f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, \mathbf{s}_c) \tag{B.12}$$

Nous comparons notre modèle à plusieurs modèles de l'état de l'art sur plusieurs jeux de données de référence tels que CUB [154], SUN [114]) et AwA2 [161]. Le protocole expérimental est essentiellement le même que celui de Xian *et al.* [161], qui proposent une comparaison équitable de nombreux modèles de ZSL avec des paramètres expérimentaux similaires, et dont les résultats font office de référence dans la littérature récente [161, 162, 152, 18, 28]. En particulier, nous utilisons les mêmes métriques pour mesurer les performances: la précision par classe $\mathcal{A}^{\text{pc}}$ ou $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ pour le contexte ZSL, et la moyenne harmonique $\mathcal{H}$ de $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ et $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ pour le contexte GZSL. Une synthèse de la signification des différentes métriques est disponible dans le tableau 1.5.

Le tableau 2.1 présente les résultats dans un contexte ZSL standard, où les instances de test appartiennent à des classes non vues et les classes candidates sont constituées de exclusivement de classes non vues afin que le score rapporté soit $A_{\mathcal{U}\to\mathcal{U}}$. Les deux variantes du modèle surpassent tous les modèles non génératifs sur deux jeux de données sur trois. Fait intéressant, le modèle final surpasse également les approches génératives sur deux ensembles de données et est le modèle le plus performant en moyenne.

Nous réalisons également une étude d'ablation afin d'évaluer l'impact des composants individuels (marge sémantique souple, normalisation partielle et poids de représentativité) de l'approche proposée. Les résultats sont présentés dans le tableau 2.2. Les trois composants fonctionnent bien ensemble : leur impact combiné est meilleur que la somme de leurs impacts marginaux.

Le tableau 2.3 contient les résultats dans un contexte GZSL, où les exemples de test (et donc les classes candidates) peuvent appartenir à des classes vues ou non vues. Nous mesurons $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ et $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ ainsi que leur moyenne harmonique $\mathcal{H}$. Comme observé dans [163] et [24], pour les approches non génératives il y a généralement un fort déséquilibre en faveur des classes vues : $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$ est significativement supérieur à $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$, ce qui pénalise le score final $\mathcal{H}$. À l'inverse, les modèles génératifs souffrent généralement beaucoup moins de ce déséquilibre des performances entre les classes vues et non vues,

car les exemples générés à partir de classes non vues sont le plus souvent utilisés en plus des exemples des classes vues pour entraîner des classifieurs supervisés standard.

B.3 Déséquilibre entre les classes vues et non vues dans un contexte d'apprentissage zéro-shot généralisé

Dans cette section, nous proposons un processus simple pour combler l'écart de performances entre les classes vues et non vues dans un contexte GZSL mis en évidence dans le tableau 2.3. Ce processus est basé sur l'idée qu'une petite dégradation de $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ pourrait être à l'origine d'une grande amélioration de $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$.

Chao *et al.* [24] ont proposé un mécanisme simple, appelé « empilement calibré » (*calibrated stacking*), dans lequel une pondération γ est introduite pour contraindre le modèle à prédire les classes non vues plus souvent lorsque $\gamma > 0$, ou les classes vues plus souvent lorsque $\gamma < 0$:

$$\hat{y} = \underset{c\in\mathcal{C}}{\operatorname{argmax}}\ f(\mathbf{x}, \mathbf{s}_c) - \gamma \mathbb{1}[c \in \mathcal{C}^{\mathcal{S}}] \tag{B.13}$$

Cependant, dans [24], aucune valeur spécifique n'est attribuée à γ. Au lieu de cela, toutes les valeurs possibles de γ de $-\infty$ à $+\infty$ sont utilisés pour estimer l'impact sur $\mathcal{A}_{\mathcal{U}}$ et $\mathcal{A}_{\mathcal{S}}$, et pour tracer une courbe paramétrique avec la valeur de $\mathcal{A}_{\mathcal{U}}$ sur l'axe x et la valeur de $\mathcal{A}_{\mathcal{S}}$ sur l'axe y (figure 2.7). L'aire sous cette courbe (*area under seen-unseen curve*) est utilisée comme métrique. Il est ici important de souligner que ce processus est appliqué *a posteriori* à un modèle de ZSL déjà entraîné afin d'évaluer ses performances. En particulier, l'impact de γ est mesuré sur le jeu de données de *test*. Par conséquent, ce processus n'est pas directement applicable pour équilibrer les performances du modèle entre les classes vues et non vues.

Par contraste, nous proposons d'employer un processus similaire, c'est-à-dire une pondération γ permettant d'ajuster la compatibilité estimée en fonction de la nature de chaque prototype $\mathbf{s}_c$, afin d'équilibrer $\mathcal{A}_{\mathcal{U}}$ et $\mathcal{A}_{\mathcal{S}}$ *sans avoir accès à l'ensemble de test*. Nous appelons ce processus le processus *calibration*. Cela nécessite de sélectionner une valeur de γ appropriée en utilisant uniquement l'ensemble d'apprentissage, ce qui est effectué avec une validation croisée spécifique au contexte de GZSL.

En GZSL, une fraction (généralement 20% [163]) des exemples des classes vues (les classes dans les ensembles d'apprentissage ou de validation) n'est pas utilisée pour l'entraînement ou la validation

et est conservée pour la phase de test afin d'évaluer $\mathcal{A}_\mathcal{S}$. Nous appelons cet ensemble *ensemble de test vu*. Ici, le terme *vu* désigne le fait que ces éléments appartiennent aux classes vues, et non le fait que ces éléments aient été vus par le modèle.

Afin de pouvoir sélectionner les hyperparamètres par validation croisée dans un contexte GZSL, nous gardons 20% supplémentaires de l'ensemble d'entraînement restant, et nous utilisons ces échantillons comme instances de classes vues lors de la validation. Nous appelons cet ensemble *ensemble de validation vu*. De cette façon, il est possible d'évaluer l'impact des hyperparamètres à la fois sur $\mathcal{A}_\mathcal{U}$ et $\mathcal{A}_\mathcal{S}$ sur l'ensemble de validation, sans utiliser aucune instance des ensembles de test vus et non vus. Tous ces différents ensembles et partitionnements sont illustrés dans la figure 2.8.

Il est maintenant possible de déterminer la valeur optimale de γ dans l'équation (B.13) avec le processus suivant. (1) L'ensemble de données est partitionné comme expliqué précédemment et comme illustré par la partie inférieure de la figure 2.8. Un modèle ZSL standard est entraîné en utilisant les exemples de l'ensemble d'apprentissage vu. (2) Sa performance de ZSL mesurée par $\mathcal{A}_{\mathcal{U}\to\mathcal{U}}$ peut être évaluée sur l'ensemble de validation non vu, de sorte que les hyperparamètres peuvent être sélectionnés de manière adéquate. (3) L'ensemble de validation non vu peut également être utilisé pour mesurer $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$; d'autre part, l'ensemble de validation vu peut être utilisé pour mesurer $\mathcal{A}_{\mathcal{S}\to\mathcal{C}}$. On peut alors tester différentes valeurs de γ et mesurer leur impact sur $\mathcal{A}_\mathcal{U}$ et $\mathcal{A}_\mathcal{S}$. Nous soulignons qu'aucun réentraînement du modèle n'est nécessaire, car γ n'affecte que la phase de prédiction d'un modèle entraîné (équation (B.13)). La valeur optimale de γ peut enfin être sélectionnée de manière à maximiser la moyenne harmonique $\mathcal{H}$ de $\mathcal{A}_\mathcal{U}$ et $\mathcal{A}_\mathcal{S}$, ou toute autre métrique de GZSL pertinente.

(4) Le modèle de ZSL est ensuite réentraîné en utilisant l'ensemble d'apprentissage vu, l'ensemble de validation vu et l'ensemble de validation non vu comme nouvel ensemble d'apprentissage. (5) Les compatibilités de classe $f(\mathbf{x}_n, \mathbf{s}_c)$ sont évaluées pour toutes les instances de test $\mathbf{x}_n$, dans l'ensemble de test non vu ainsi que dans l'ensemble de test vu, pour tous les prototypes de classe $\mathbf{s}_c$. (6) Pour les prototypes de classe $\mathbf{s}_c$ des classes vues $c \in \mathcal{C}^\mathcal{S}$, la valeur de γ sélectionnée précédemment est soustraite des compatibilités correspondantes conformément à l'équation (B.13) pour prédire $\hat{y}$. (7) $\mathcal{A}_\mathcal{U}$ et $\mathcal{A}_\mathcal{S}$ peuvent finalement être mesurés respectivement sur l'ensemble de test non vu et sur l'ensemble de test vu pour obtenir le score final $\mathcal{H}$.

Comme le montre la figure 2.9, les hyperparamètres optimaux en GZSL peuvent être différents des hyperparamètres optimaux en ZSL. Ceci peut être expliqué par une décomposition de l'erreur en un

compromis biais-variance spécifique au GZSL, ce qui est illustré dans la figure 2.10. Nous proposons la procédure suivante pour sélectionner la valeur optimale de régularisation λ d'un modèle de régression ridge (équations (B.17) et (B.18), section B.5) ou tout autre ensemble d'hyperparamètres dans un contexte GZSL : nous répétons le protocole décrit précédemment pour sélectionner la pondération de calibration γ, et nous prenons la valeur de λ qui donne le meilleur résultat pour la moyenne harmonique $\mathcal{H}$ de $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ et $\mathcal{A}_{\mathcal{U}\to\mathcal{C}}$ sur l'ensemble de validation *après* avoir soustrait γ des compatibilités des classes vues comme dans l'équation (B.13). Le reste du processus est identique.

Les résultats expérimentaux mesurant l'impact des processus proposés de calibration et de sélection des hyperparamètres sont disponibles dans le tableau 2.7, et montrent une amélioration significative pour la plupart des modèles évalués sur tous les jeux de données. Dans le tableau 2.8, nous comparons les résultats GZSL de ces modèles au modèle propos section B.2, avec la calibration et les hyperparamètres sélectionnés spécifiquement pour la tâche de GZSL. Le procédé proposé permet d'augmenter le score $\mathcal{H}$ de notre modèle de plus de 18 points en moyenne. Notre modèle surpasse alors toutes les approches non génératives sur deux jeux de données sur trois, et surpasse également les approches génératives « en moyenne ».

B.4 Représentations sémantiques non supervisées

Dans cette section, nous nous concentrons principalement sur l'impact des représentations sémantiques dans un contexte d'apprentissage zéro-shot à grande échelle. Lorsque le nombre de classes est important, il peut être fastidieux de concevoir des représentations sémantiques de haute qualité à l'aide d'attributs définis manuellement. Par exemple, le jeu de données CUB contient 312 attributs par classe, et les performances des modèles de ZSL diminuent très rapidement avec le nombre d'attributs utilisé, comme illustré figure 3.2. Il est ainsi rarement envisageable de créer manuellement des centaines d'attributs pour chacune des milliers de classes d'un jeu de données à grande échelle comme ImageNet. Dans un contexte de grande échelle, il est donc courant d'utiliser les plongements lexicaux des noms des classes pour obtenir des prototypes sémantiques, ce qui requiert un effort d'annotation quasiment nul. Ceci s'explique par le fait que les modèles de plongement de mots sont entraînés de manière *non supervisée*, et ne nécessitent donc pas d'annotations préalables. Par exemple, dans l'approche Word2vec [102], l'objectif du *skipgram* vise à prédire les mots présents dans le contexte d'un mot donné. Si nous considérons un corpus de L phrases, où la $l^{\text{ème}}$ phrase est constituée de T_l

mots $\{w_1, \ldots, w_{T_l}\}$, l'objectif est de minimiser

$$-\sum_{l=1}^{L}\sum_{t=1}^{T_l}\sum_{\substack{-S\leq i\leq S\\ i\neq 0}} \log p(w_{t+i}|w_t) \tag{B.14}$$

Ici, S est la taille de la fenêtre contextuelle et indique à quel point deux mots w_i, w_j doivent être proches l'un de l'autre pour être considérés comme faisant partie de leurs contextes respectifs. La probabilité $p(w_i|w_t)$ dans l'équation (B.14) est généralement estimée à l'aide d'un réseau neuronal entièrement connecté (*fully connected*) à une couche cachée, l'activation de la couche cachée étant utilisée comme le plongement du mot w. La représentation sémantique $\mathbf{s}_c$ pour la classe c peut par la suite être obtenue en utilisant le plongement lexical correspondant à son nom, où la moyenne des différents plongements dans le cas d'une classe dont le nom est constitué de plusieurs mots. Cependant, les performances avec ce type de représentation sémantique sont généralement bien inférieures à celles des attributs conçus manuellement, comme représenté figure 3.2.

Une explication possible de cette différence est que les corpus de texte habituellement utilisés pour entraîner les modèles de plongements lexicaux ne contiennent pas suffisamment d'informations visuelles, comme illustré figure 3.1. On peut ainsi faire l'hypothèse que les modèles de plongements lexicaux entraînés sur des corpus avec une nature plus visuelle pourraient conduire à des embeddings mieux adaptées à la tâche de ZSL. Nous proposons ainsi de créer de tels ensembles de données.

Nous utilisons l'API Flickr pour collecter des balises (*tags*) définies par les utilisateurs : étant donnée une requête q constituée de mots-clés tels que *« tigre »*, l'API renvoie une liste d'images et de métadonnées associées. Nous nous intéressons à trois champs dans les métadonnées associées à chaque image : (1) Le titre, qui est une description de l'image définie par l'utilisateur, par exemple *"Amur tiger chilling in the water "*[3] pour un résultat de la requête *« tigre »*. (2) Une liste de balises associées à l'image également définies par l'utilisateur, par exemple *« tigre »', « sumatra », « faune »* ou encore *« tigre », « orange », « zoo »* pour la requête *« tigre »*. (3) L'identifiant de l'utilisateur, afin que chaque image puisse être associée à un utilisateur unique.

Nous nous basons sur deux approches pour déterminer les requêtes à utiliser pour créer le corpus complet. Dans la première approche, nous utilisons des concepts génériques de Wikipedia pour créer la collection Flickr-Wikipedia, ou $\mathbf{fl_{wiki}}$. Pour sélectionner des concepts génériques communs, les pages

[3] *"Amur tiger chilling in the water"* dans la version non traduite, associé à la requête *"tiger"*.

Wikipédia sont triées en fonction de leur nombre de liens entrants dans le corpus Wikipédia, et les 120 000 premiers concepts sont conservés. Les titres de ces pages sont utilisés comme requêtes pour collecter des métadonnées en utilisant Flickr comme décrit précédemment. Dans une seconde approche appelée $\mathbf{fl}_{\mathbf{cust}}$, nous utilisons les noms des classes du jeu de données de ZSL pour collecter des métadonnées spécifiques à la tâche.

Chaque collection se base sur Q concepts et les requêtes associées $\mathcal{Q} = \{q_1, \ldots, q_Q\}$, $Q \leq 120000$. Pour chaque requête q, nous obtenons une liste de M_q résultats avec les métadonnées associées $\mathcal{M}_q = \{m_1, \ldots, m_{M_q}\}$, $M_q \leq 5000$. Chaque résultat (métadonnées) m se compose d'une liste de mots $\mathcal{W}_m = \{w_1, \ldots, w_{T_m}\}$, ainsi que d'un identifiant utilisateur id_m associant l'auteur id_m de l'image et des métadonnées m avec un utilisateur unique u_{id_m}. Les T_m mots sont les mots constituant le titre et les balises.

Dans nos collections de texte, contrairement aux corpus standard tels que Wikipedia, l'ordre des mots $\mathcal{W}_m = \{w_1, \ldots, w_{T_m}\}$ dans chaque résultat (métadonnées) m est arbitraire. Par conséquent, la fenêtre de contexte de taille fixe de la formulation habituelle de l'objectif skipgram (équation (B.14)) n'est pas adaptée : deux mots apparaissant dans le même contexte, par exemple dans les mêmes métadonnées m, ne sont pas nécessairement proches l'un de l'autre $\{w_1, \ldots, w_{T_m}\}$.

Au lieu d'utiliser une fenêtre de taille fixe, nous considérons que deux mots w_i et w_j apparaissent dans le même contexte si les deux apparaissent dans l'ensemble de mots $\mathcal{W}_m$ des métadonnées m. L'objectif skipgram de l'équation (B.14) peut donc être adapté ainsi :

$$-\sum_{q \in \mathcal{Q}} \sum_{m \in \mathcal{M}_q} \sum_{\substack{(w_i, w_j) \\ w_i, w_j \in \mathcal{W}_m,\ i \neq j}} \log p(w_i | w_j) \tag{B.15}$$

Cette formulation est équivalente à extraire toutes les paires de mots (w_i, w_j) telles que w_i, w_j appartiennent au même $\mathcal{W}_m$ comme dans l'équation (3.3), et à créer un corpus dont les phrases sont constituées des paires de tels mots. L'entraînement du modèle de plongement peut alors être effectué avec l'objectif skipgram de l'équation (3.1). La même méthode peut être appliquée pour apprendre des plongements avec d'autres méthodes telles que GloVe [117] ou FastText [14]. Cela présente l'avantage de permettre l'utilisation des implémentations standard de ces différents modèles.

Un problème pouvant survenir avec des jeux de données constitués de balises définies par l'utilisateur est celui du « balisage de masse » (*bulk-tagging*) [111] : ce terme décrit l'action d'utilisateurs attribuant

les mêmes balises ou descriptions à de grands ensembles de photos. Ce phénomène peut parfois biaiser les modèles de langage obtenus à partir de Flickr ou d'autres sites communautaires [121, 111]. Pour éviter ce problème, nous mettons en place un « filtrage par utilisateur » (*user filtering*), c'est-à-dire que nous appliquons la règle selon laquelle une paire de mots (w_i, w_j) est prise en compte au plus une fois par utilisateur Flickr distinct. Concrètement, cela se traduit par l'ajout d'une paire (w_i, w_j) dans le corpus d'entraînement une unique fois pour chaque utilisateur.

Nous comparons les résultats obtenus avec l'approche proposée aux résultats obtenus avec des plongements standard formés sur des corpus de texte génériques, pour une variété de modèles de ZSL. Nous comparons également avec deux approches basiques : dans l'approche **wiki**, nous entraînons les plongements lexicaux avec les méthodes standard Word2vec, GloVe et FastText sur le corpus Wikipédia habituel. Dans l'approche **clue**, nous entraînons les plongements sur un sous-ensemble de la collection ClueWeb12 [21, 20], ayant pour but de correspondre à un contenu textuel plus visuellement connoté. Nous extrayons pour cela les données associées aux images référencées dans l'ensemble de données, en récupérant les métadonnées des attributs HTML *title* et *alt* correspondants.

Puisque nous nous intéressons aux résultats à grande échelle, nous menons principalement nos expériences sur l'ensemble de données ImageNet [33]. Il a été proposé dans [40] d'utiliser les 1000 classes d'ILSVRC [132] en tant que classes d'entraînement, et les 20 841 autres en tant que classes de test non vues. Cependant, il a été récemment mis en évidence qu'un biais structurel peut survenir dans ce cadre, et permettre à un modèle « trivial » exploitant spécifiquement ce biais de surpasser la plupart des modèles de ZSL existants [53]. Pour cette raison, nous adoptons le protocole d'évaluation proposé par Hascoet *et al.* [53], qui considèrent les mêmes classes d'entraînement que [40] mais utilisent 500 classes avec un biais structurel minimal pour les tests.

Pour avoir un aperçu de l'écart existant entre les attributs définis manuellement et les plongements obtenus de façon non supervisée, nous menons également des expériences sur des benchmarks plus petits, sur lesquels la tâche de ZSL est généralement accomplie avec des attributs manuels. Nous utilisons ainsi également les ensembles de données CUB [154] et AwA2 [161]. Les attributs manuels habituels de CUB et AwA2 comportent respectivement 312 et 85 dimensions. À l'exception des attributs, nous adoptons le protocole expérimental de Xian *et al.* [161], et plus précisément leurs partitionnements apprentissage-test proposés (*"proposed splits"* pour ces deux jeux de données.

Les principaux résultats pour le jeu de données à grande échelle ImageNet sont rapportés dans le tableau 3.1. Les meilleurs résultats sont la plupart du temps obtenus avec l'approche fl_{cust} et FastText. Les prototypes de classe obtenus avec cette approche surpassent significativement les résultats précédents dans un contexte grande échelle.

Nous rapportons également des résultats pour les jeux de données à plus petite échelle CUB et AwA2 dans les tableaux 3.2 et 3.3. Ces résultats sont moins pertinents car des attributs manuels existent pour ces jeux de données, mais apportent tout de même des informations intéressantes. Il est important de garder à l'esprit que ces résultats sont produits à l'aide de prototypes obtenus de façon *non supervisée* lors de la comparaison avec les résultats possibles avec des attributs manuels. Sur CUB, les meilleurs résultats sont obtenus avec les embeddings appris sur la collection fl_{cust} pour les trois modèles de plongement et surpassent les embeddings précédents. De façon intéressante, il ne semble pas y avoir de tendance marquée sur AwA2. Il s'avère que les performances atteignables avec des prototypes non supervisés sur AwA2 sont déjà assez proches des performances avec des attributs manuels, comme visible figure 3.2. La méthode proposée n'est donc pas en mesure d'apporter une amélioration significative sur ce jeu de données.

Afin de mesurer l'impact du filtrage des utilisateurs, nous comparons les résultats obtenus avec et sans cette étape avec l'approche fl_{cust} appliquée à FastText, le modèle de plongement le plus performant sur l'ensemble de données ImageNet pour différents modèles ZSL. Ces résultats sont reportés dans le tableau 3.4 et confirment que limiter l'impact de chaque utilisateur sur le corpus d'apprentissage a un impact positif significatif sur les performances finales du modèle de ZSL.

Pour quantifier les performances des prototypes conçus manuellement par rapport aux prototypes « non supervisés », nous retirons progressivement des attributs des prototypes de classe de CUB et AwA2. Nous commençons avec la liste complète des attributs, et nous supprimons aléatoirement certains attributs tout en mesurant le score ZSL en résultant. Les scores sont mesurés avec un modèle linéaire (équations (B.17) et (B.18)) en raison de ses bons résultats, de sa robustesse et de sa simplicité. Les résultats sont visibles figure 3.2, les points bleus représentant le score moyen pour un nombre fixe d'attributs fabriqués à la main. Sur CUB, il y a encore une marge d'amélioration substantielle. La différence entre les prototypes supervisés et annotés manuellement n'est pas aussi prononcée sur AwA2.

B.5 Utilisation de descriptions courtes en tant que représentations sémantiques

Nous avons obtenu des résultats intéressants dans un contexte grande échelle en créant et en exploitant des corpus avec une connotation plus visuelle, ce qui permet de créer des plongements plus performants à partir des noms de classe. Cependant, sur certains datasets tels que CUB, les résultats avec une telle approche sont encore nettement inférieurs à ceux pouvant être obtenus avec des attributs définis manuellement. Nous défendons que cette approche peut être limitée pour la reconnaissance visuelle fine. Par exemple, même avec des plongements créés à partir d'un corpus plus « visuel », il n'est pas évident qu'il soit possible d'apprendre la différence entre un terrier australien et un terrier irlandais, deux classes visibles figure 3.5 et présentes dans le jeu de données ImageNet, juste à partir des embeddings de leurs noms de classe.

Une solution idéale pourrait être l'utilisation de courtes phrases en langage naturel pour décrire chaque classe. Une telle solution prendrait moins de temps que de fournir des attributs exhaustifs pour chaque classe et pourrait être plus informative visuellement que les plongements de mots appris à partir de corpus de texte génériques. Des exemples de telles descriptions courtes pourraient être « petite race de terrier grisâtre à poil dur d'Australie » et « race de taille moyenne avec un pelage brun raide développée en Irlande » pour les classes respectives *« terrier australien »* et *« terrier irlandais »*[4]. Dans cette section, nous proposons différentes approches pour utiliser de courtes descriptions en langage naturel dans le cadre de l'apprentissage zéro-shot.

L'approche la plus simple pour obtenir un prototype sémantique à partir d'une courte description consiste à utiliser la moyenne des plongements des mots constituant la description. Nous appelons cette approche $\mathbf{Def}_{\mathbf{average}}$. Cependant, comme illustré figure 3.8, tous les mots d'une description ne sont généralement pas d'importance égale. Nous nous intéressons donc à l'utilisation de mécanismes d'attention : nous construisons l'embedding d'une phrase comme une moyenne pondérée des embeddings de ses mots, de sorte que les mots plus importants contribuent davantage à l'embedding résultant. Nous envisageons deux manières d'y parvenir. Sauf indication contraire, nous utilisons un modèle de régression linéaire régularisé de l'espace sémantique vers l'espace visuel (équations (B.18)

[4]Ces deux classes sont présentes dans ImageNet et les descriptions mentionnées sont traduites des définitions WordNet correspondantes.

et (B.17)) pour évaluer les prototypes de classe résultants.

Dans l'approche appelée $\mathbf{Def}_{\mathbf{visualness}}$, nous essayons d'estimer à quel point un mot est pertinent d'un point de vue visuel. Pour un mot w_i donné, nous collectons les $M_i \leq 100$ images les plus pertinentes de Flickr. Nous obtenons des représentations visuelles $\{\mathbf{r}_1^i, \ldots, \mathbf{r}_M^i\}$ pour les images collectées M en utilisant un ResNet-101 [55] pré-entraîné. Nous faisons l'hypothèse que pour les mots à forte connotation visuelle, les représentations visuelles des images collectées correspondant à ce mot sont proches les unes des autres. Nous mesurons ainsi la distance moyenne des vecteurs $\mathbf{r}_m^i$ au vecteur moyen $\bar{\mathbf{r}}^i$ pour obtenir l'inverse de la pertinence visuelle v_i. Des exemples de mots avec une pertinence visuelle élevée ou faible sont représentés figure 3.6. Ces exemples tendent à confirmer que cette pertinence visuelle estimée est raisonnable.

Étant donné une définition de T mots dont les embeddings correspondants sont $\{\mathbf{v}_1, \ldots, \mathbf{v}_T\}$, nous appliquons ensuite la fonction *softmax* sur les pertinences visuelles correspondantes $\{v_1, \ldots, v_T\}$ pour obtenir une représentation $\mathbf{s}$ de la phrase à partir de la moyenne pondérée des plongements, donnant ainsi plus de poids aux mots connotés visuellement. Comme l'échelle initiale des distances moyennes ou pertinences visuelles inverses est arbitraire, une température τ est introduite dans le softmax, de sorte que le plongement de phrase résultant est

$$\mathbf{s} = \sum_{t=1}^{T} \frac{\exp(v_t/\tau)}{\sum_{k=1}^{T} \exp(v_k/\tau)} \mathbf{v}_t \tag{B.16}$$

τ est considéré comme un hyperparamètre et sa valeur est sélectionnée par validation croisée sur l'ensemble de validation. Une illustration des pondérations résultantes pour quelques phrases est présentée dans la figure 3.8 (à gauche).

Dans l'approche $\mathbf{Def}_{\mathbf{attention}}$, nous visons à apprendre à prédire la pertinence v^i du mot w^i à partir de son plongement $\mathbf{v}^i \in \mathbb{R}^K$ tel que $v^i = \mathbf{w}^\top \mathbf{v}^i$ où $\mathbf{w}$ sont des paramètres appris. L'équation (B.16) peut ensuite être appliquée pour créer un prototype sémantique à partir d'une définition. Différentes manières pourraient être envisagées pour apprendre les paramètres $\mathbf{w}$. Une approche simple pourrait consister à initialiser aléatoirement $\mathbf{w}$ avec les paramètres $\mathbf{W}$ du modèle linéaire (équation (B.17)), puis à estimer les pertinences $v^i = \mathbf{w}^\top \mathbf{v}^i$ pour chaque mot w^i, à construire les prototypes de classe $\mathbf{s}_c$ pour chaque classe c en utilisant v^i et l'équation (3.7), à calculer $\mathbf{T} = (\mathbf{t}_1, \ldots, \mathbf{t}_N)^\top$ avec $\mathbf{t}_n = \mathbf{s}_{y_n}$, et enfin à évaluer la fonction de coût du modèle linéaire :

$$\frac{1}{N} \|\mathbf{X} - \mathbf{T}\mathbf{W}\|_2^2 + \lambda \|\mathbf{W}\|_2^2 \tag{B.17}$$

Nous pourrions alors utiliser une descente de gradient (à l'aide d'une rétropropagation) pour mettre à jour $\mathbf{w}$ et $\mathbf{W}$ jusqu'à leur convergence. Cependant, nous pouvons profiter de l'existence d'une solution analytique à l'équation (B.17) et procéder comme suit : nous initialisons aléatoirement $\mathbf{w}$ et construisons les prototypes de classe $\mathbf{s}_c$ ainsi que $\mathbf{T}$ en utilisant l'équation (B.16) comme précédemment. Nous estimons alors directement $\mathbf{W}$ en utilisant la solution analytique à l'équation (B.17)

$$\mathbf{W} = (\mathbf{T}^\top \mathbf{T} + \lambda N \mathbf{I}_K)^{-1} \mathbf{T}^\top \mathbf{X} \tag{B.18}$$

et utilisons cette estimation de $\mathbf{W}$ pour calculer le coût dans l'équation (B.17). Nous rétro-propageons ensuite le gradient et effectuons la descente du gradient sur $\mathbf{w}$ uniquement, la valeur de $\mathbf{W}$ étant estimée avec l'équation (B.18) à chaque itération. Nous répétons ce processus jusqu'à la convergence. Les poids d'attention résultants de quelques phrases sont illustrés figure 3.8 (à droite).

Plutôt que de combiner des plongements de mots dans une définition pour former un prototype de classe unique avec une dimension fixe, une autre solution peut consister à adapter les approches de ZSL existantes pour permettre l'utilisation d'un prototype de classe composé de plusieurs parties avec la même dimension fixe. Nous proposons d'explorer cette approche en adaptant le coût de triplet évoqué section B.2 afin de permettre l'utilisation de plusieurs prototypes par classe.

Au lieu d'avoir une classe c représentée par un vecteur sémantique unique $\mathbf{s}_c$, nous considérons maintenant que chaque classe c est représentée par un ensemble de T_c prototypes $\{\mathbf{s}_c^1, \dots, \mathbf{s}_c^{T_c}\}$, avec $\mathbf{s}_c^t \in \mathbb{R}^K$. De tels prototypes peuvent par exemple être les embeddings $\{\mathbf{v}_1, \dots, \mathbf{v}_{T_c}\}$ des mots constituant une courte description de la classe. Le coût de triplet de l'équation (B.1) peut alors être adapté pour que par exemple, pour la classe correcte y, seuls les P embeddings $\{\mathbf{s}_c^{p_1}, \dots, \mathbf{s}_c^{p_P}\}$ avec la compatibilité $f(\mathbf{x}, \mathbf{s}_y^p)$ la plus élevée soient pris en compte. En écrivant $\mathcal{T}_P = \{p_1, \dots, p_P\}$ les index des « top-P » plongements, le coût de triplet devient

$$[m + f(\mathbf{x}, \mathbf{s}_c) - \frac{1}{P} \sum_{p \in \mathcal{T}_P} f(\mathbf{x}, \mathbf{s}_y^p)]_+ \tag{B.19}$$

Nous pouvons remarquer que dans cet exemple, nous considérons toujours qu'il existe un prototype unique $\mathbf{s}_c$ pour les classes incorrectes $c \neq y$ au lieu d'un ensemble de représentations textuelles $\{\mathbf{s}_c^1, \dots, \mathbf{s}_c^T\}$. Un tel prototype peut être obtenu simplement en faisant la moyenne des embeddings de mots de telle sorte que $\mathbf{s}_c = \frac{1}{T} \sum_{t=1}^{T} \mathbf{s}_c^t$, de façon similaire à la précédente approche $Def_{average}$.

Plus généralement, nous pouvons considérer de la même manière les Q plongements avec la compatibilité la plus élevée pour les classes incorrectes $c \neq y$. En adaptant le coût de triplet de l'équation (B.19) et en additionnant les coûts sur l'ensemble d'apprentissage, cela se traduit par l'objectif suivant :

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{\substack{c\in\mathcal{C}^{\mathcal{S}}\\ c\neq y_n}}\left[m+\frac{1}{Q}\sum_{q\in\mathcal{T}_Q} f(\mathbf{x}_n,\mathbf{s}_c^q)-\frac{1}{P}\sum_{p\in\mathcal{T}_P} f(\mathbf{x}_n,\mathbf{s}_{y_n}^p)\right]_+ \tag{B.20}$$

Nous pouvons remarquer que si P et Q sont égaux au nombre de mots dans chaque définition, cela équivaut à faire la moyenne des compatibilités avec tous les plongements lexicaux. Si f est en outre linéaire par rapport à $\mathbf{s}$, cette équation est équivalente à la fonction de coût de triplet standard des équations (B.1) et (B.2). Pour effectuer des prédictions avec un modèle entraîné avec l'objectif de l'équation (B.20), nous pouvons à nouveau utiliser la moyenne des R plongements lexicaux avec la compatibilité la plus élevée, et prédire la classe non vue c avec la moyenne la plus élevée :

$$\hat{y}=\underset{c\in\mathcal{C}^{\mathcal{U}}}{\operatorname{argmax}}\frac{1}{R}\sum_{r\in\mathcal{T}_R} f(\mathbf{x},\mathbf{s}_c^r) \tag{B.21}$$

.

Les plongements de phrases obtenus à partir des deux approches $Def_{visualness}$ et $Def_{attention}$ peuvent être comparés aux prototypes de classe standard obtenus à partir du plongement des noms de classe. Nous appelons cette approche l'approche **Classname**. Nous menons nos expériences sur le jeu de données à grande échelle ImageNet [33], en utilisant les définitions WordNet [103] correspondant aux *synsets* associés aux classes en tant que courtes descriptions. Nous utilisons le même protocole expérimental que dans la section B.4, c'est-à-dire le même que dans [53]. Nous évaluons les performances avec les trois modèles de plongement Word2vec, GloVe et FastText. Nous menons également des expériences avec le modèle de plongement Elmo [118], qui permet d'obtenir des embeddings contextuels.

Les résultats sont présentés dans le tableau 3.6. L'approche de base $Def_{average}$ fonctionne moins bien que l'approche habituelle *Classname*. Les mécanismes d'attention apportent une légère amélioration, avec des résultats comparables pour les deux approches $Def_{visualness}$ et $Def_{attention}$, mais les performances restent nettement inférieures à celles de l'approche *Classname*. Ces résultats sont cohérents avec ceux rapportés dans [54], dans lequel les plongements obtenus à partir des définitions WordNet sont également à l'origine de résultats nettement inférieurs à ceux obtenus avec les noms de classe. Les

résultats obtenus avec Elmo sont étonnamment moins bons, y compris avec l'attention, alors que l'on pouvait s'attendre à ce que l'attention soit plus efficace ici dans la mesure où les plongements d'Elmo peuvent être considérés comme incluant le rôle d'un mot dans une phrase.

Nous comparons les résultats obtenus avec l'approche multi-prototype de l'équation (B.20) avec une approche standard de coût de triplet (équations (B.1) et (B.2)) dans laquelle les plongements de mots sont moyennés pour un seul prototype $\mathbf{s}_c = \frac{1}{T}\sum_{t=1}^{T}\mathbf{s}_c^t$ par classe c. Les résultats sont présentés dans le tableau 3.7. Le modèle multi-prototype obtient de meilleurs résultats que DeViSE lorsque nous utilisons les lemmes distincts d'un synset comme multi-prototypes. Cependant, les résultats du modèle multi-prototype avec des multi-prototypes composés de mots issus de définitions sont encore médiocres, malgré des scores de compatibilité qui semblent raisonnables d'un point de vue qualitatif comme illustré figure 3.9). Les scores atteints avec cette approche sont également inférieurs à ceux des approches d'attention.

Les résultats décevants des deux approches peuvent s'expliquer en partie par le fait que certaines des définitions de WordNet n'incluent pas vraiment d'informations pertinentes pour décrire une classe. Par exemple, pour les deux classes de test *« tourterelle »* et *« tourterelle tigrine »*, les descriptions correspondantes sont respectivement « n'importe quelle espèce de colombe sauvage d'Europe, d'Afrique ou d'Asie » et « petite colombe australienne »[5]. Cependant, nous émettons l'hypothèse que dans d'autres cas, il peut malgré tout y avoir des informations supplémentaires intéressantes dans ces définitions par rapport aux seuls noms de classe.

Nous étudions donc une approche très simple pour combiner ces sources d'information : étant donné $\mathbf{s}_1$ et $\mathbf{s}_2 \in \mathbb{R}^K$ deux représentations sémantiques obtenues avec des approches différentes, par exemple les approches *Classname* et $Def_{average}$, nous créons un prototype combiné $\mathbf{s}$ via une combinaison convexe des deux prototypes paramétrée par un scalaire $\mu \in [0, 1]$ telle que $\mathbf{s} = \mu\mathbf{s}_1 + (1-\mu)\mathbf{s}_2$. Nous utilisons cette idée pour combiner des prototypes *Classname* avec des plongements de phrases qui, contrairement aux approches multi-prototypes, ont l'avantage de fournir facilement une seule représentation à dimension fixe d'une phrase. Nous appelons par exemple **Classname + Def$_{\text{visualness}}$** la combinaison du prototype *Classname* avec le prototype $Def_{visualness}$. Puisque des résultats récents indiquent que les relations hiérarchiques et graphiques entre les classes contiennent des informations

[5]Les descriptions originales des classes *turtledove* et *Australian turtledove* sont *"any of several old world wild doves"* et *"small Australian dove"*.

intéressantes [53, 60], dans l'approche **Classname + Parent**, nous combinons le prototype obtenu en utilisant un nom de classe avec le prototype obtenu à partir de sa classe parent dans la hiérarchie WordNet. Nous définissons de la même manière l'approche *Classname + Def*$_{visualness}$ *+ Parent* comme la combinaison du prototype *Classname + Def*$_{visualness}$ avec le prototype de sa classe mère, ce dernier étant également obtenu par une combinaison de plongements de noms de classe et de définitions.

Nous sélectionnons la valeur de μ conjointement avec les hyperparamètres du modèle en utilisant la validation croisée, sauf lors de la combinaison d'un prototype avec le prototype de sa classe parent. Dans ce dernier cas, la validation croisée a tendance à produire des valeurs très incohérentes et instables, ce qui est cohérent avec les résultats de [4]. Lors de la combinaison d'un prototype avec son prototype parent, nous imposons relativement arbitrairement $\mu = 0,75$.

Les résultats expérimentaux sont présentés dans le tableau 3.8. La combinaison de *Classname* avec les approches *Def* apporte des scores nettement meilleurs que l'un ou l'autre séparément. L'utilisation d'informations parentes en plus des prototypes *Classname* améliore les résultats par rapport aux prototypes enfants seuls, ce qui est cohérent avec [53] où les meilleures méthodes utilisent des relations hiérarchiques entre les classes. Le même effet est observé lors de l'utilisation des informations parentales avec l'approche *Classname + Def*$_{visualness}$. Enfin, nous fournissons des résultats supplémentaires pour différents modèles de ZSL dans le tableau 3.9 pour l'approche la plus performante, l'approche *Classname + Def*$_{visualness}$ *+ Parent* avec les plongements FastText, et nous rapportons les résultats de [53] pour comparaison. Les performances sont considérablement améliorées pour tous les modèles.

B.6 Conclusion et perspectives

Dans cette thèse, nous nous avons étudiés l'apprentissage zéro-shot en tant que moyen de réduire le besoin d'annotations humaines pour entraîner des modèles de reconnaissance visuelle. Nous nous sommes en particulier concentrés sur le contexte plus réaliste d'apprentissage zéro-shot généralisé, dans lequel les classes de test peuvent être à la fois vues et non vues, ainsi que sur le contexte plus difficile de zéro-shot à grande échelle, dans lequel les prototypes sémantiques sont obtenus de manière non supervisée. Dans une première partie, nous avons fait valoir que les méthodes de classement font habituellement plusieurs hypothèses implicites potentiellement préjudiciables. Nous avons pro-

posé d'adapter la formulation habituelle du coût de triplet afin que ces méthodes puissent prendre en compte les relations inter et intra-classe. Dans une deuxième partie, nous avons proposé un processus simple pour combler l'écart entre les performances sur les classes vues et non vues, dont ces méthodes souffrent fréquemment dans un contexte d'apprentissage zéro-shot généralisé. Dans notre évaluation expérimentale, la combinaison de ces contributions a permis au modèle que nous proposons d'égaler ou de surpasser les performances des méthodes génératives, tout en étant moins restrictif en pratique. Dans une troisième partie, nous nous sommes concentrés sur les représentations sémantiques utilisées dans un cadre d'apprentissage zéro-shot à grande échelle. Dans ce contexte, les informations sémantiques proviennent généralement de plongements lexicaux des noms de classe. Nous avons défendu l'idée selon laquelle les plongements habituels souffrent d'un manque de contenu à connotation visuelle dans les corpus d'entraînement. Nous avons donc proposé de nouveaux corpus de textes davantage connotés visuellement ainsi qu'une méthode pour adapter les modèles d'apprentissage de plongements à ces corpus. Enfin, dans une quatrième et dernière partie, nous avons proposé de compléter ces représentations non supervisées par de courtes descriptions en langage naturel, dont la production ne nécessite qu'un effort minimal par rapport à des attributs exhaustifs. Ces différentes contributions ont permis d'obtenir des résultats significativement améliorés par rapport à l'état de l'art antérieur.

Il pourrait être intéressant d'explorer plus en détail plusieurs des contributions proposées. Par exemple, les méthodes que nous proposons pour utiliser de courtes descriptions comme représentations sémantiques sont assez simples pour le moment, même si elles conduisent déjà à une augmentation des scores lorsqu'elles sont combinées avec d'autres types de représentations. Cette simplicité est en partie due à la quantité limitée de données d'entraînement disponibles, car nos modèles n'avaient accès qu'à un millier de phrases courtes lors de la phase d'entraînement. Avec un ensemble de données d'apprentissage plus grand, il est probable que d'autres approches plus complexes puissent mener à de meilleurs résultats. Même si les scores actuels sont sans doute encore trop faibles pour de nombreux cas d'utilisation pratiques, nous restons optimistes concernant la tâche d'apprentissage zéro-shot à grande échelle. Nous espérons que nos contributions émuleront d'autres recherches au sujet des représentations sémantiques des classes. En effet, cet aspect semble être un élément important pour les scénarios d'apprentissage zéro-shot à grande échelle, et pourtant très peu de travaux portent sur ce sujet. De la même manière que la reconnaissance zéro-shot a bénéficié des progrès de la vision

par ordinateur en général, il est également probable que les progrès dans différents domaines tels que le traitement du langage naturel puissent bénéficier à cette tâche. À plus long terme, un modèle de reconnaissance zéro-shot efficace à grande échelle et basé entièrement sur des prototypes de classe non supervisés représenterait un atout majeur en matière d'approches frugales en termes de données, et pourrait ainsi grandement contribuer à démocratiser l'utilisation de la vision par ordinateur et de l'apprentissage automatique.

le cnam

Yannick LE CACHEUX
Toward more practical zero-shot learning

le cnam

Résumé : Cette thèse porte sur la reconnaissance visuelle « zero-shot », qui vise à classifier des images de catégories non rencontrées par le modèle pendant la phase d'apprentissage. Après avoir classé les méthodes existantes en trois grandes catégories, nous défendons l'idée que les méthodes dites de classement se basent habituellement sur plusieurs hypothèses implicites préjudiciables. Nous proposons d'adapter leur fonction de coût pour leur permettre d'intégrer des relations inter et intra-classe. Nous proposons également un processus permettant de diminuer l'écart entre les performances sur les classes vues et non vues dont souffrent fréquemment ces méthodes. Dans notre évaluation expérimentale, ces contributions permettent à notre modèle d'égaler ou surpasser les performances des méthodes génératives, tant en étant moins restrictif. Dans un second temps, nous nous intéressons aux représentations sémantiques utilisées dans un contexte d'application à grande échelle. Dans ce contexte, l'information sémantique provient généralement de plongements lexicaux des noms de classe. Nous soutenons que les plongements habituels souffrent d'un manque de contenu visuel dans les corpus servant à leur apprentissage. Nous proposons donc de nouveaux corpus de texte davantage connotés visuellement, ainsi qu'une méthode permettant d'adapter les modèles de plongement à ces corpus. Nous proposons en outre de compléter ces représentations non supervisées par de courtes descriptions en langage naturel, dont la production ne requiert qu'un effort minimal comparé à des attributs génériques.

Mots clés : Apprentissage zero-shot, reconnaissance visuelle, apprentissage automatique

Abstract : This thesis focuses on zero-shot visual recognition, which aims to recognize images from unseen categories, i.e. categories not seen by the model during training. After categorizing existing methods into three main families, we argue that ranking methods habitually make several detrimental implicit assumptions. We propose to adapt the usual formulation of the hinge rank loss so that such methods may take inter and intra-class relations into account. We also propose a simple process to address the gap between accuracies on seen and unseen classes, from which these methods frequently suffer in a generalized zero-shot learning setting. In our experimental evaluation, the combination of these contributions enables our proposed model to equal or surpass the performance of generative methods, while being arguably less restrictive. In a second part, we focus on the semantic representations used in a large-scale zero-shot learning setting. In this setting, semantic information customarily comes from word embeddings of the class names. We argue that usual embeddings suffer from a lack of visual content in training corpora. We thus propose new visually oriented text corpora as well as a method to adapt word embedding models to these corpora. We further propose to complete unsupervised representations with short descriptions in natural language, whose generation requires minimal effort when compared to extensive attributes.

Keywords : zero-shot learning, image recognition, machine learning