



## University of Genoa

---

DITEN - Electronics and Telecommunication Engineering and  
Naval Architecture Department

PhD in Science and Technology for Electronic and  
Telecommunication Engineering - Cycle XXXIV

*Curriculum in Computational vision, recognition and machine  
learning*

---

## Learning with Unavailable Data: Generalized and Open Zero-Shot Learning

PhD Thesis

*PhD Candidate:*  
Federico Marmoreo

*Supervisor:*  
Prof. Vittorio Murino

*Co-Supervisor:*  
Dr. Jacopo Cavazza

*Coordinator of the PhD Course:*  
Prof. Maurizio Valle



## Abstract

The field of visual object recognition has seen a significant progress in recent years thanks to the availability of large-scale annotated datasets. However, labelling a large amount of data is difficult and costly and can be simply infeasible for some classes due to the long-tail instances distribution problem.

Zero-Shot Learning (ZSL) is a framework that consider the case in which for some of the classes no labeled training examples are available to train the model. To solve the problem a multi-modal source of information, the class (semantic) embeddings, is exploited to extract knowledge from the available classes, the seen classes, and recognize novel categories for which the class embeddings is the only information available, namely, the unseen classes.

To directly targeting the extreme imbalance in the data, in this thesis, we first propose a methodology to improve synthetic data generation for the unseen classes through their class embeddings. Second, we propose to generalize the Zero-Shot Learning framework towards a more competitive and real-world oriented scenario. Thus, we formalize the problem of Open Zero-Shot Learning as the problem of recognizing seen and unseen classes, as in ZSL, while also rejecting instances from unknown categories, for which neither visual data nor class embeddings are provided. Finally, we propose methodologies to not only generate unseen categories, but also the unknown ones.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Contributions of the Thesis . . . . .	5
1.3	Outline of the Thesis . . . . .	7
1.4	Publications . . . . .	8
<b>2</b>	<b>Background and Related Work</b>	<b>9</b>
2.1	Problem Definition . . . . .	9
2.1.1	Class Embeddings . . . . .	10
2.1.2	Visual Features . . . . .	11
2.2	Evaluation Metrics . . . . .	13
2.3	Generative Methods . . . . .	15
2.4	Closed-Set and Open-Set Assumptions . . . . .	18
2.5	Literature Review . . . . .	20
<b>3</b>	<b>Decoupled Feature Generation</b>	<b>24</b>
3.1	Context . . . . .	24
3.2	Related Work . . . . .	28
3.3	Decoupled Feature Generation . . . . .	30
3.3.1	Notation and Problem Definition . . . . .	30

3.3.2	Our Proposed Architecture: DecGAN . . . . .	30
3.3.2.1	Unconditional Branch. . . . .	32
3.3.2.2	Conditional Branch. . . . .	33
3.3.2.3	Cross Branch. . . . .	34
3.3.3	Training Methodology . . . . .	35
3.3.4	Implementation Details . . . . .	36
3.4	Experiments . . . . .	36
3.4.1	Datasets and Benchmarks . . . . .	36
3.4.2	Ablation Study . . . . .	37
3.4.2.1	The Impact on Performance of each Training Stage.	38
3.4.2.2	The Effect of the Decoupled Feature Generation.	39
3.4.3	Comparison with the State-of-the-Art in transductive Gen- eralized Zero-Shot Learning . . . . .	41
3.5	Conclusions . . . . .	44
<b>4</b>	<b>Open Zero-Shot Learning</b>	<b>45</b>
4.1	Context . . . . .	45
4.2	Related Work . . . . .	49
4.3	Problem Formulation . . . . .	50
4.3.1	Open Zero-Shot Learning evaluation protocol. . . . .	51
4.3.2	OSZL datasets. . . . .	52
4.3.3	Error metrics. . . . .	52
4.4	Unknown Feature Generation . . . . .	55
4.4.1	Direct Unknown Generation (DUG) . . . . .	55
4.4.2	Semantic Based Unknown Generation (SBUG) . . . . .	56
4.4.2.1	VAcWGAN . . . . .	57
4.4.2.2	Implementation details . . . . .	58
4.4.2.3	Unknown Generation. . . . .	59

## CONTENTS

---

4.5	Experiments . . . . .	60
4.5.1	Open Zero-Shot Learning through Unknown Generation . . . . .	63
4.6	Conclusions and Future Work . . . . .	65
<b>5</b>	<b>Conclusion</b>	<b>66</b>
	<b>References</b>	<b>85</b>
<b>A</b>	<b>Supplementary Material for Chapter 4</b>	<b>86</b>
A.1	Recent Advances in Visual-Language Models . . . . .	86
A.2	Proposed Splits for Open Zero-Shot Learning . . . . .	89

# List of Figures

1.1	Example of attribute based classification. If the model is able to learn the attributes “ <i>equine</i> “ and “ <i>black and white pattern</i> “ on the seen categories, transferring this knowledge on the unseen domain, is able to recognize, based on these attributes, a Zebra, even if it was trained without Zebra examples. To achieve knowledge transfer, the attributes have to be shared and discriminative across the classes. . . . .	2
2.1	Representation of GAN architecture. The Generator $G$ takes as input a vector of random variables $z$ , producing a synthetic examples $\tilde{x}$ . The Discriminator $D$ takes as input synthetic examples $\tilde{x}$ and real data $x$ assigning a <i>score</i> representing if they are real or fake. GAN can be extended to a conditional model if both, $G$ and $D$ , take as input a condition factor $c$ . . . . .	16
3.1	Our proposed DecGAN architecture is composed of two cross-connected branches consisting of two GANs: unconditional and conditional. The unconditional branch (yellow) is composed of generator $G^0$ and discriminator $D^0$ , and the conditional branch (light blue) is composed of generator $G^c$ and discriminator $D^c$ . An additional cross-branch (violet) is composed of $G^c$ and $D^0$ . . . . .	31

3.2	DecGAN training is performed in 3 stages. Stage 1 performs an alternate training on the conditional and the unconditional branch using seen data. Stage 2 uses the unconditional branch to fine-tune $G^0$ using unseen data to improve the structured prior $\mathbf{s}$ . Finally, Stage 3 carries out the fine-tuning of $G^c$ , feeding the cross-branch with unseen data. . . . .	35
3.3	Baseline architecture. <i>Left</i> : a visualization of the architectural design of the baseline. <i>Right</i> : similarly to our proposed DecGAN, we adopt a staged training to optimize the baseline. . . . .	40
4.1	<b>Open Zero-Shot Learning</b> , a framework where we aim at classifying seen and unseen classes (for which no visual data of the latter is given) while also rejecting ( <i>i.e.</i> , refusing to take any decision on) unknown classes. Neither visual data nor class embeddings are available for unknown classes. . . . .	45
4.2	<b>The proposed pipeline for Open Zero-Shot Learning.</b> We synthesize visual descriptors from seen and unseen classes, using a Generative Adversarial Network (GAN). We also learn how to perform unknown generation and synthesize descriptors (represented by $\circ\circ\circ\circ$ ), even for the unknown classes, and better precondition a classifier in classifying seen/unseen and reject unknown, with the usage of Openmax [1]. . . . .	47
4.3	Using VAcWGAN, we generate unknown class embeddings (in a transformed semantic space) from which, in turn synthetic unknown visual features can be generated. . . . .	59



4.4 Precision and Recall of CLSWGAN [2] combined with either Softmax or Openmax [1] on the AWA dataset. Openmax has been tuned using different tail sizes (ranging from 2 - darkest red - to 10 - lighter red). When adopting state-of-the-art solutions (like [1]) to cope with the unknown, we argue that the joint presence of unseen classes (which we do not have to forget) prevents Openmax to reliably rejecting the unknown - as it appears to be able to if we remove unseen classes (see [1]). We perceive this as evidence of the challenges related to Open Zero-Shot Learning. . . . . 61

# List of Tables

3.1	Statistics of considered datasets: number of seen classes $\#\mathcal{Y}^s$ (training + validation), unseen classes $\#\mathcal{Y}^u$ , dimension of attribute (att) annotation and sentences (stc) extracted features. . . . .	37
3.2	We assess the impact on performance of the presence/absence of each stage of the training pipeline of our DecGAN model. We report top-1 accuracy on seen classes $\mathbf{a}_s$ and unseen classes $\mathbf{a}_u$ and their harmonic mean $\mathbf{H}$ . Best $\mathbf{H}$ values are highlighted in bold. All results are reported by averaging accuracies over 5 different runs.	38
3.3	The effect of decoupling the feature generation stage. We compare the decoupled approach of DecGAN to the not decoupled baseline. We report top-1 accuracy on seen classes $\mathbf{a}_s$ and unseen classes $\mathbf{a}_u$ and their harmonic mean $\mathbf{H}$ . Best $\mathbf{H}$ values are highlighted in bold. All results are reported by averaging accuracies over 5 different runs.	39

3.4	Results in Generalized Zero-Shot Learning. We report top-1 accuracy on seen classes $\mathbf{a}_s$ and unseen classes $\mathbf{a}_u$ and their harmonic mean $\mathbf{H}$ . First and second best values are highlighted in bold and italic, respectively, for $\mathbf{H}$ . Inductive (I) and Transductive (T) methods are reported. †: results not reported because of the usage of different class embeddings, which are not comparable. Our results are presented by averaging performance scores over 5 different runs. . . . .	43
4.1	Baseline methods for Open Zero-Shot Learning evaluated on their capability of rejecting unknown (treated as a separated class for which precision $P_\Omega$ , recall $R_\Omega$ , and F1 $F1_\Omega$ scores can be computed. We also focus here on classifying unseen classes, reporting the average F1 score $F1_u$ over them, while also reporting the per-class F1 score for two exemplar classes whose performance is above the mean (6th, 7th columns, marked in green), and two classes that are below it (8th, 9th columns, marked in red). We observe that, generically, Openmax achieves high recall and low precision. The softmax is not capable of rejecting, therefore $P_\Omega = R_\Omega = F1_\Omega = 0\%$ .	62
4.2	Direct and semantic unknown generation (DUG and SBUG) for the WAcWGAN model (check Sec. 4.4.2). . . . .	64
4.3	The effect of direct unknown generation (DUG) applied on three benchmark methods: CSLWGAN [2], tf-VAEGAN [3] and CEZSL [4] on the proposed OZSL splits for AWA [5], CUB [6], FLO [7] and SUN [8] datasets. . . . .	64

# Chapter 1

## Introduction

In this Chapter, we will give an introduction of faced problems and overview of our research, pointing out the original contributions.

### 1.1 Overview

The huge increase of visual data availability of the last twenty years made a revolution in computer vision, started back in the 1940s, possible. The will to understand and simulate how neurons of the brain work led to the idea of artificial neural networks, but the theory was faster than applications. The power of the neural networks is in their ability to autonomously learn *patterns* and extract *features* from the data, largely outperforming algorithms based on hand-crafted features and reaching human level performances in many computer vision tasks. On the downside, neural networks are eager for data to learn, thus, to prove their capabilities and use them in real-world applications, the scientific community had to wait for the possibility to collect and analyze a big corpus of annotated data. Therefore, it was possible to prove their capabilities and largely outperform the competition only in the last years.




SEEN		UNSEEN
Horse	Rosalia Funebris	Zebra
		
Equine: yes	Equine: no	Equine: yes
Black and white pattern: no	Black and white pattern: yes	Black and white pattern: yes

Figure 1.1: Example of attribute based classification. If the model is able to learn the attributes “*equine*” and “*black and white pattern*” on the seen categories, transferring this knowledge on the unseen domain, is able to recognize, based on these attributes, a Zebra, even if it was trained without Zebra examples. To achieve knowledge transfer, the attributes have to be shared and discriminative across the classes.

However, collecting and labelling a massive amount of data is not only difficult and expensive, but sometimes unfeasible [9]. In fact, while for some classes can be easy to collect a massive amount of visual data, for example is very easy to collect pictures of an actor as Brad Pitt, for other classes can be very difficult to collect a balanced amount of data, for example for a common person like an average PhD student, and for other classes can be impossible, like the famous criminal D.B. Cooper.

For this reason, different captivating research topics in machine learning and computer vision focus on limited data source availability. For example, in Domain Adaptation problem [10; 11; 12] the categories to be classified are available during training, but there could be limited or not at all labeled data for the target

domain of application. For example, for self-driving car applications, it can be difficult or expensive to collect data for the full possible weather, lighting or context conditions. Besides, in Few-Shot Learning problem [13; 14], while some categories are largely represented by training examples, for some other categories very few training examples are available.

In this thesis, we investigate an extreme case of the aforementioned issues: the Zero-Shot Learning problem. Differently from a conventional fully supervised paradigm, in which each single category is assumed to be (evenly) represented by a set of annotated visual data, Zero-Shot Learning allows this assumption to hold only for a restricted set of *seen* categories. The goal is then to recognize a disjoint set of target *unseen* categories, for which labeled data are not available. Thus, Zero-Shot Learning is the problem of recognizing novel categories, even if a classifier has not been directly trained on them [15; 16], and to solve it we need to develop machine learning methodologies that can work in these limited data constraints.

To achieve these goals, the inspiration comes from the ability of human beings to generalize in categorization, identifying completely new classes when provided with high-level descriptions. For example, if the equines are already known, from the phrase “*equines with distinctive black-and-white striped coats*” is possible to recognize a zebra even if the zebra has been never seen previously. Thus, to transfer knowledge from seen to unseen categories, auxiliary information can be exploited, and it is actually typically adopted in the form of either manually-defined attributes [16] or features extracted from a neural network from textual input (as the distributed word embeddings [17]). To perform classification, this auxiliary information is required to be shared and discriminative across the classes, and we refer to it as class embeddings (see Figure 1.1).

Classical approaches to solve Zero-Shot Learning either can firstly perform

class embedding prediction followed by class prediction, or can directly learn a compatibility function between visual features and class embeddings. These methods have shown compelling performances in the original Zero-Shot Learning framework, that consider only unseen classes for evaluation, but fail to balance performances between seen and unseen classes.

In fact, in order to define a more realistic and competitive scenario, *generalized* Zero-Shot Learning has been proposed. Generalized Zero-Shot Learning considers both the unseen and the seen classes for evaluation, thus it is important to address the extreme data imbalance problem in order to overcome the natural bias towards predicting the class for which more training examples are available.

Latest competitive approaches in Generalized Zero-Shot Learning directly address this imbalance by generating a synthetic dataset for the unseen classes through the class embeddings. These methods build upon the possibility of mimicking the human brain in hallucinating a mental imagery of a certain unseen category, while reading a textual description of it. State-of-the-art methods rely on neural networks based architecture that, conditioned on the class embeddings, generate corresponding class examples. Ideally, the generative process for Generalized Zero-Shot Learning can be schematized in the three main following steps.

1. Train a generative process to generate visual features based on the class embeddings. Seen data are used to learn the relation between class embeddings and the visual features since we have the labels.
2. Use the unseen class embeddings to generate labeled synthetic data for the unseen classes. This is possible since the class embeddings are shared across the classes.
3. Train a classifier using the real labeled data for the seen classes and the

synthetic labeled data for the unseen ones.

## 1.2 Contributions of the Thesis

In this thesis, targeting the Generalized Zero-Shot Learning, we introduce a novel, more effective, feature synthesis method to balance, hence improving the training process. The goal of the generative process is to generate synthetic descriptors indistinguishable from a pool of pre-trained real features. To achieve this goal the generative process has to solve two tasks: first, capturing the data distribution of the seen and unseen domains and second, translating the semantic information into visual patterns.

Hence, differently from prior work, we propose to separately solve the two tasks by *decoupling* the feature generation stage to better control the generation process. We implement the decoupled feature generation with a novel architecture, named DecGAN. With this architecture we can separate the two tasks into two different branches, having a better control on each of them. Additionally we can take advantage, when available, of unlabeled unseen data by combining the two branches. With this additional step we alleviate the domain-shift problem [3; 18; 19; 20] when moving from generating the seen domain to generating the unseen one. We validate our architecture through an extensive ablation study and outperform with it the state-the-art methods on different Generalized Zero-Shot Learning benchmark datasets.

On the one hand, with DecGAN we work to improve the generative process for the competitive Generalized Zero-Shot Learning framework, on the other hand we work to propose an even more general and competitive setup. With this setup we continue the process of extending Zero-Shot Learning to more real-world oriented scenarios started with Generalized Zero-Shot Learning.



In Generalized Zero-Shot Learning, both the seen and the unseen classes are considered for evaluation, but under the assumption of knowing in advance the full set of classes and their class embeddings. We claim that this assumption (namely the closed-set assumption) is still a limitation for Zero-Shot Learning methods in real/world applications. In fact, while it is reasonable to assume that we can describe all the seen classes with the class embeddings, it seems too constraining to know and be able to describe all the classes for which we have no labeled visual data. In fact, collecting rich and expressive class embeddings could be difficult and costly.

To overcome the closed-set assumption and moving to the open-set scenario (that is, we consider a possible infinite set of classes at inference time), we introduce a third type of classes. Other than the seen, for which we have visual data and class semantic descriptors, the unseen, for which we have only class embeddings, we add the unknown, for which we have neither the visual data nor the semantic class embeddings. Thus, we extend Generalized Zero-Shot Learning to Open Zero-Shot Learning, where inference has to be jointly performed over seen, unseen and unknown classes in order to classify seen and unseen, and reject unknown ones.

As contributions, we formally define the Open Zero-Shot Learning problem, we provide publicly available benchmark datasets and we propose evaluation metrics to allow fair and reproducible comparison across different algorithmic solutions tackling Open Zero-Shot Learning. Our evaluation metrics extend the ones used in Generalized Zero-Shot Learning to better handle the open set scenario. We also evaluate baselines based on Generalized Zero-Shot Learning state-of-the-art methods in the open set scenario. Additionally, we propose a methodology to directly synthesize unknown visual features and we also propose our novel idea to synthesize unknown class embeddings implementing it through our architecture

VAcWGAN.

### 1.3 Outline of the Thesis

In this section, we provide an overview of the content of the thesis.

**Chapter 2: Related Work** In this chapter, we formally define the Zero-Shot Learning problem and its different setups providing an introduction of the concepts and methods used in the next chapters. We also present a survey of the related work in Zero-Shot Learning, discussing how these works relate to the contributions of this thesis.

**Chapter 3: Decoupled Feature Generation** In this chapter, we present the idea and the implementation of the decoupled feature generation, together with a detailed ablation study to dissect the effect of our proposed decoupling approach, while demonstrating its superiority over the related state-of-the-art.

**Chapter 4: Open Zero-Shot Learning** In this chapter, we formalize the Open Zero-Shot Learning problem, introducing evaluation protocols, error metrics and benchmark datasets. We also suggest tackling the Open Zero-Shot Learning problem by proposing the idea of performing unknown feature generation.

**Chapter 5: Conclusion** In this chapter, we summarize the contribution of our thesis.

## 1.4 Publications

- Marmoreo Federico, Jacopo Cavazza and Vittorio Murino. (2021). “Transductive Zero-Shot Learning by Decoupled Feature Generation.” *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 3108-3117.
- Marmoreo Federico, Julio Ivan Davila Carrazco, Vittorio Murino and Jacopo Cavazza. (2021). “Learning without Seeing nor Knowing: Towards Open Zero-Shot Learning.” *Under review*.

# Chapter 2

## Background and Related Work

In this Chapter we will formally introduce the Zero-Shot Learning problem and the generative models that will be used in Chapters 3 and 4. We also present the most relevant work and recent developments in Zero-Shot Learning.

### 2.1 Problem Definition

Zero-Shot Learning is a classification problem that considers two disjoint sets of classes: the *seen* classes  $\mathcal{S}$  and the *unseen* ones  $\mathcal{U}$  ( $\mathcal{S} \cap \mathcal{U} = \emptyset$ ). Differently from a classical classification problem, a training set of labeled visual data is available only for the seen classes.

We distinguish between *inductive* Zero-Shot Learning, where training visual data are not available at all for the unseen classes, and *transductive* Zero-Shot Learning, where for the unseen classes unlabeled data are available.

The task is to correctly classify visual examples belonging to the unseen domain. To achieve this goal, we use an auxiliary source of information that lets us perform a knowledge transfer from the seen domain to the unseen one. We refer to this auxiliary source of information as the *class embeddings*.

### 2.1.1 Class Embeddings

The class embeddings play a key role in Zero-Shot Learning since they provide class level information that lets us recognize the novel classes.

Widely used class embeddings in Zero-Shot Learning are the *attribute* embeddings. With attributes visual properties of the class can be described, as for example the color or the shape. Among the benefits of these embeddings, first they are shared across the classes and thus let us perform the knowledge transfer from the seen to the unseen domain, second they are very intuitive to understand and more resemble the human approach in recognizing novel categories based on visual properties. On the downside, they are not only required to be shared across classes, but to perform classification they are also required to be discriminative across them, thus they have to be sufficiently fine-grained for the domain of interest. More importantly, attributes are obtained through human annotation, thus can be difficult or costly to obtain. For example the attributes for the Caltech-UCSD Birds-200-2011 [6], a standard benchmark dataset in Zero-Shot Learning, attributes are obtained selecting 312 visual properties (for example “*wing shape*” and “*wing colour*”), and have been obtained with crowdsourcing.

As a solution to avoid costly human annotation, *word* embeddings can be used. Word embeddings are representations of words, usually in the form of a real-valued vector, such that the words that are similar in meaning are projected close in the vector space [21; 22; 23]. Commonly used word embeddings in Zero-Shot Learning are word2vec [22; 24]. Word2vec are computed training a neural network on a very large corpus of text, specifically can be trained either with a continuous bag-of-words architecture, that predicts the current word based on the context, or the Skip-gram model, that predicts surrounding words given the current word. Word2vec are inexpensive to compute and had been show to capture high semantic information of words and also able to translate the semantic rela-

tion between words into mathematical operations, for example  $vector("Athens") - vector("Greece") + vector("Norway") = vector("Oslo")$ . However, they could not capture visual properties and consequently lead to poor performances in Zero-Shot Learning.

To capture visual features from a textual description *text* embeddings have been proposed [25]. If visual descriptions of the images are available, a long short-term memory recurrent neural network [26; 27] is used to learn a sentence representation that is close to the visual features representation (see Section 2.1.2). These descriptors improve over the word2vec, but require a more difficult training and the construction of the text description dataset.

Very recent work in jointly learning visual and textual embeddings may help in overcoming current class embeddings limitations in future work and a discussion is presented in Appendix A.1.

### 2.1.2 Visual Features

As we anticipated in Chapter 1.1, neural networks are widely used to extract visual features representations from an image and achieve state-of-the-art performances. The class of artificial neural networks used to analyze visual imagery are the convolutional neural networks.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In image classification, the input consists of the image, represented as a matrix, while the output is the predicted class. The hidden layers, that involve convolutional operations, are named convolutional layers. Convolutional layers are composed of kernels (or filters), represented by matrices and non-linear functions called activation functions. If the input matrix has dimensions (height) x (width) x (depth), Frobenius inner products are computed shifting the kernel matrix along the height and width dimensions of the input matrix. Then the

output of the Frobenius inner products is used as input for the following activation function. This operation generates a matrix that is used as input for the next layer. The output of the hidden layers (that is a real valued matrix or vector) is referred to as visual features, with the addition of the adjective deep to refer to the higher, or deeper, layers of the neural network.

Convolutional neural networks often also include pooling layers. A pooling layer locally aggregates the features by the max value or averaging them.

Intuitively, at the first hidden layer, passing the kernels all over the image they learn local simple properties of the image, as for example lines, that are represented by a feature, while going deeper (that is in the following layers) into the network these features became always less local and more abstract, as for example shapes or faces, representing high level information about the image. Differently from hand-crafted kernels, the kind of features represented by the network are not defined, but is the neural network that learns them during the training process in order to minimize the cost function.

Pioneering work in convolutional neural networks has been proposed by Yann LeCun in 1988 with the LeNet-5 architecture [28] and the following year, 1989, with the back-propagation algorithm to perform optimization [29]. LeNet-5 was proposed to classify handwritten numbers and has been applied by several banks to recognize cheques, but was limited in the possibility to be applied in more difficult problem or higher resolution images due to the need of a large amount of annotated data and training resources to be applied to bigger problems. So the interest in convolutional neural networks started rising in the 2000s when big corpus of data started to be available and GPU parallel implementations were proposed. A proof of convolutional neural networks potential arrived in 2012, when AlexNet architecture, an improvement over the original LeNet-5, won the ImageNet Large Scale Visual Recognition Challenge [30]. Later, much effort has

been made to train deeper and more complex convolutional neural networks, as for example GoolgleNet[31], VGG [32], ResNet [33] and Inception [34].

To take advantage of convolutional neural networks also on smaller dataset, that is a big corpus of data is not available, a common practice is to use pre-trained networks. In fact these networks learn different kinds of features depending on the layer, from low-level features to high level features going deeper in the networks as we mentioned before. The usage of big corpus of data, with high variability in the domain, generally leads to very general features. In fact, while the final layers are used for classification, and thus can be specific for the domain, the previous layers focus on learning very general patterns to analyze the images. For these reasons, when using a pre-trained network, generally only the last layers are fine-tuned, or learned from scratch, on the new domain.

For fair comparison in Zero-Shot Learning, generally, visual features used for bench-marking different methodologies are fixed and obtained from the top pooling layer of ResNet101 [33] deep convolutional network. The network has been trained on the large scale dataset ImageNet [35] and not fine-tuned [17].

## 2.2 Evaluation Metrics

As we have already introduced in Section 1.1 we distinguish between two main frameworks, that are Zero-Shot Learning and the more real-world oriented scenario Generalized Zero-Shot Learning [17].

**Zero-Shot Learning.** In Zero-Shot Learning, for evaluation, only unseen classes are considered. To balance the performance on each class and consequently take into account the possible long-tail distribution of the classes, that is some classes are less represented from others, the accuracy of the classifier is averaged over the accuracy on every class and we refer to it as mean-per-



class-accuracy  $\mathcal{A}$ .

In detail, given  $u \in \mathcal{U}$  an unseen class,  $R_u$  the accuracy, or recall, of the classifier for  $u$  (that is the number of the correctly classified examples divided by the number of test example for  $u$ ) the mean-per-class-accuracy on the unseen classes  $\mathcal{A}_u$  is given by

$$\mathcal{A}_u = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} R_u, \quad (2.1)$$

where  $|\mathcal{U}|$  is the total number of unseen classes.

**Generalized Zero-Shot Learning.** Generalized Zero-Shot Learning evaluation is performed on both, the seen and unseen classes. To balance the performance on the two disjoint sets of classes, the harmonic mean  $\mathcal{H}$  of the mean-per-class-accuracy on the unseen classes  $\mathcal{A}_u$  and the mean-per-class-accuracy on the seen classes  $\mathcal{A}_s$  is computed. In detail,

$$\mathcal{H} = 2 \frac{\mathcal{A}_s \mathcal{A}_u}{\mathcal{A}_s + \mathcal{A}_u}, \quad (2.2)$$

where  $\mathcal{A}_u$  is defined in equation (2.1) and similarly,

$$\mathcal{A}_s = \frac{1}{|\mathcal{S}|} \sum_{u \in \mathcal{S}} R_s, \quad (2.3)$$

where  $R_s$  is the recall for the seen class  $s \in \mathcal{S}$  and  $|\mathcal{S}|$  the total number of seen classes.

## 2.3 Generative Methods

As we anticipated in Section 1.1, recent works in Zero-Shot Learning take advantage of deep generative methods to produce synthetic data for the unseen classes. Generative Adversarial Networks (GANs) [36; 37] are one of the most used and flexible artificial neural network architectures for data generation, that showed state of the art performances in different generative problems as conditional image generation [38; 39; 40; 41; 42].

GANs are an implicit generative model. This means that the distribution of the data is not directly defined, but it is directly learned by the model from the training example. To learn the distribution, two different adversarial neural networks are used, a Generator  $G$  and a Discriminator  $D$ . The goal of the Generator is to produce data as *realistic* as possible, while the task of the Discriminator is to distinguish between real and synthetic data.

The Generator takes as input random noise, that is vectors composed of random variables, and produces as output synthetic data examples. The Discriminator takes as input synthetic and real examples and assigns a score to each of them, representing if they are real or fake (see Figure 2.1). As for example in the initial formulation [36], the output is a value between 0 and 1 representing the probability of being a real example.

To perform optimization, a two player min-max game is performed. As in the case of [36; 37], while the Discriminator tries to maximize the probability of assigning the correct label (*real* or *synthetic*) to the examples, the Generator is trained to fool the Discriminator by minimizing the difference between the *true* label and the value assigned by the Discriminator. Playing this game, the Generator implicitly defines a distribution of the data, obtained by directly producing data examples from the non-observable latent variables used as input.

GANs showed to be a very powerful and flexible architecture to model very

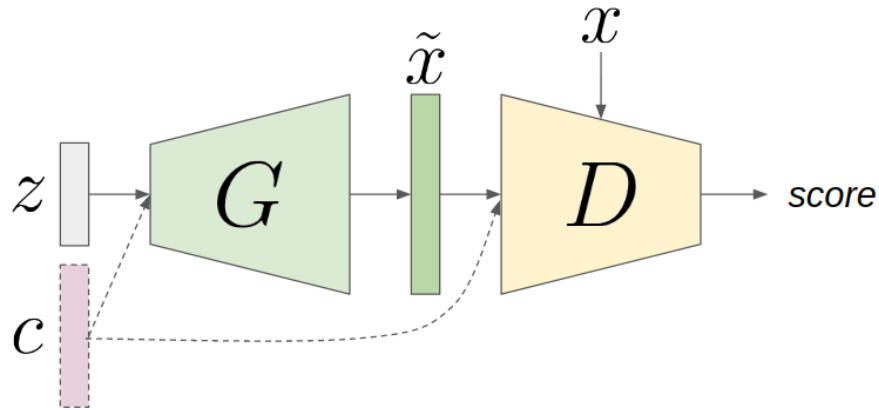


Figure 2.1: Representation of GAN architecture. The Generator  $G$  takes as input a vector of random variables  $z$ , producing a synthetic examples  $\tilde{x}$ . The Discriminator  $D$  takes as input synthetic examples  $\tilde{x}$  and real data  $x$  assigning a *score* representing if they are real or fake. GAN can be extended to a conditional model if both,  $G$  and  $D$ , take as input a condition factor  $c$ .

complex data distributions, becoming a state-of-the-art paradigm for image generation. However their training can be unstable and could suffer from mode collapsing (that is the Generator produces synthetic examples with very limited variety). As shown by [43], as the Discriminator gets better in distinguishing between real and synthetic examples, the gradient gets closer to 0, providing no useful information to update the Generator. Thus we need to carefully balance the training and the updates on the Generator and the Discriminator.

To address this issue, Wasserstein GAN has been proposed [44; 45]. While in GANs, in their original formulation [36], the Jensen-Shannon divergence between the distribution defined by the Generator and the real one is minimized, in Wasserstein GANs their Earth-Mover (or Wasserstein-1) distance is minimized. With the usage of the Earth-Mover distance, we can avoid to carefully balance the training of the Generator and the Discriminator, since the Generator can be trained until optimality and high quality gradients can be used to train the Generator [44].

Another common family of artificial neural networks used for data generation is Variational Autoencoders (VAEs) [46]. VAEs start from a different theoretical point of view compared to GANs, trying to explicitly approximate the intractable data distribution.

VAEs are composed of two neural networks: an Encoder and a Decoder. The Encoder maps the input example into the latent space, associating to each image normal distributions in the latent space. A vector sampled from the distributions, is used as input for the Decoder, which tries to reconstruct the input example. To regularize the latent space, the normal distributions are constrained to be close to a standard normal distribution (that is mean 0 and variance 1).

Optimization is performed using two terms: a reconstruction term, that is the mean square error between the input and its reconstruction and the regularization term, that is the Kulback-Leibler divergence between the encoded distribution and a standard Gaussian.

After the training the Decoder can be used in the same way of a GAN Generator, that is firstly sampling from the latent space a random input vector and secondly using it as input to generate a synthetic example.

GANs and VAEs can be extended to conditional models if both, the Generator and the Discriminator, or the Encoder and the Decoder, takes some conditioning information as input [47]. In Zero-Shot Learning we are interested in conditioning on the class embeddings. In fact, if a generative model is trained to generate class specific visual examples based on the class embeddings, it can be used to generate labeled synthetic visual features for the unseen classes reducing the extreme data imbalance. Ideally, the generative model can be trained using seen visual features and class embeddings, and then, since as described in Section 2.1.1 the class embeddings are shared across the classes, it can be conditioned on the unseen class embeddings to generate unseen visual features. Finally, having

labeled examples for all the seen and unseen classes, the final recognition stage can be solved by exploiting a simple classifier.

## 2.4 Closed-Set and Open-Set Assumptions

As we already anticipated in the introduction, in Chapter 4 we will extend the Generalized Zero-Shot Learning setup to Open Zero-Shot Learning.

Standard and Generalized Zero-Shot Learning setups, are both defined under the closed-set assumption, that is all the set of classes valuated at inference time is finite and known. We consider this assumption a strong constrain to apply Zero-Shot Learning in real-world scenarios. Indeed, it is reasonable to assume the possibility to describe with the rich content of the class embeddings all the classes for which we have training data, but could be very hard or impossible to describe with them all the other categories, for which no labeled example is provided. In fact, as we presented in Section 2.1.1, having class embeddings that are semantically rich and shared and discriminative across all the classes can be hard or impossible to achieve, in particular for classes we have never actually seen.

To overcome these limitations, we move towards the open-set setup, that is we consider a possibly infinite number of classes at inference time dividing them into three disjoint sets:

1. the *seen* classes, for which we have the labeled visual data and the class embeddings;
2. the *unseen* classes, for which do not have the labeled training data but we have the class embeddings;
3. the *unknown* classes, for which we do not have the labeled data and neither

## 2.4 Closed-Set and Open-Set Assumptions

---

the class embeddings.

We define, with Open Zero-Shot Learning, the problem of classifying the seen and the unseen classes while rejecting the unknown ones.

When moving from the closed-set assumption to the open one, the main challenge is to control the classification regions in the feature space. Generally, in a closed-set environment, the classifier is trained to assign portions of the feature space to every known class, based on the information provided during training. If no constraint is provided, the classifier divides all the feature space (that is  $\mathbb{R}^d$ , where  $d$  is the number of features) into classification regions for the known classes. This means that every unknown instance presented at inference is misclassified as one of the known ones. To avoid this phenomenon, ideally, we should *shrink* the classification region as much as possible around the regions where we see the known classes and reject everything that is outside.

To tackle this problem, classical approaches in open-set environments based on deep visual features leverage on the statistical properties of the classifier learned to measure the *confidence* of the classifier in the prediction [1; 48; 49; 50]. However, these methods only consider seen and unknown classes.

The main challenge when removing the closed-set assumption in Zero-Shot Learning, is that for the unseen classes we do not have labeled data or we do not have data at all. Thus, measuring the *confidence* of a classifier is not straightforward.

Generative approaches for Zero-Shot Learning can be used to synthetically generate features for the unseen classes and thus combining the two methodologies. However, the synthetically generated features could not necessarily own the required statistical properties of the real ones. Furthermore, while the classifier can be very confident in classifying the seen classes, it can be less confident on the unseen domain since the synthetic to real domain shift.

To tackle Open Zero-Shot Learning, benchmark methods, obtained combining the current state of the art, and new models will be discussed in Chapter 4.

## 2.5 Literature Review

In this Section we summarize the most relevant work in Zero-Shot Learning and the most popular families of approaches that have been presented throughout the last years in which Zero-Shot Learning has captured increasing interest.

The pivotal work in the Zero-Shot Learning [51] proposes direct attribute prediction. With direct attribute prediction, a first classifier predicts the attributes, that are the values of the class embeddings, then the class is predicted with the one closer to that prediction. Directly predicting the attribute can be unreliable, for example not all the attributes could be discriminative enough to perform classification or equally difficult to predict. To address this issue [52] still perform direct attribute prediction, but leveraging statistics about each attribute’s error tendencies, propose a random forest approach to train zero-shot models that explicitly accounts for the unreliability of attribute predictions for classification.

Due to the unreliability of the attributes, succeeding methods integrate the two stages, that is firstly learning the relation between the visual features and secondly performing classification, in a single method. These succeeding methods can be described as compatibility learning frameworks. The general idea of these frameworks is to project the visual features and the semantic embeddings in a common space and then perform classification based on a compatibility function between the two. The weights of the projections are learned directly with a classification loss, differently from the previous methodologies that optimize the ability of predicting the attributes. A common choice for the compatibility function is a bilinear function in the class embeddings and visual features

[53; 54; 55; 56; 57; 58]. For example, DeVISE [54] projects the visual features into the space of the class embeddings such that the model produces a higher dot-product similarity between the visual features and the correct class embeddings, while ESZSL [55] proposes a formulation of the problem based on linear relations between visual features and class embeddings that allows a simple but effective closed-form solution. Non-linear approaches to project the visual features and the class embeddings have been investigated, as for example in [59], where two VAE are used to learn aligned latent representation of the two.

Other approaches aims at describing the unseen classes as combinations of the seen classes [60; 61; 62; 63], while others approaches [64; 65] are based on dictionary-learning [66].

Even if these methods showed compelling performance in the conventional Zero-Shot Learning setup, they struggle in the more competitive Generalized Zero-Shot Learning setup. In fact, in the generalized setup the classifier has to balance the performance on both, the seen and the unseen classes, and the extreme unbalanced Zero-Shot Learning setup can lead to classifiers biased towards the seen classes. To address this issue, by generating a synthetic dataset for the unseen classes, generative approaches for Zero-Shot Learning have been proposed. [67] uses an explicit density model with tractable density, that is, the density function of the conditional probability of the visual features given the class embedding, is modeled through an exponential family of distributions. With an explicit and tractable density function is very simple to implement and the parameter estimation reduces to solving a regression problem, for which a closed-form solution exists. However, constraining the density function limits the possibility to capture all the complexity of the data. To overcome this limitation deep generative models have been exploited. In fact, in the deep generative models, presented in previous Section 2.3, is the neural network that learns the



mapping between the latent space and the visual space. Thus much effort has been spent to adapt them to the Zero-Shot Learning framework and to improve the generative capabilities on the unseen domain. f-CLSWGAN [2] uses a conditional Wasserstein GAN, with the addition of a classification loss in the training to force the network producing discriminative features. In [68] multi-modal cycle-consistency loss [69] is added to f-CLSWGAN framework, that is the semantic embedding used to generate the visual feature has to be subsequently predicted from the generated visual features, and a similar approach can be found also in [70; 71]. [72] regularizes the GAN generation by forcing the generated visual features to be close to the corresponding class cluster centroid. [4] maps both the real and the synthetic samples produced by the deep generative model into a new space, where with the usage of contrastive embedding the features are more class separated and discriminative. [73] maps the generated features in a new space to remove the redundant information from the visual features without losing the discriminative information during the GAN training. In [74] a VAE architecture is used. In [75] a conditional VAE is used together with triplet loss and center loss to increase separability between classes. [76] trains a VAE to learn disentangled feature representation in the latent space. In [77; 78; 79] a GAN and a VAE architectures are combined by sharing the Generator/Decoder.

For the transductive setup, where unseen unlabeled data are available, [80] proposes a procedure to perform label propagation [81] that, as shown in [17], can be extended to the methods based on compatibility function. But due to the shift between seen and unseen classes, the projection of the visual features and the semantic embedding in a common space may struggle when switching from the seen to the unseen domain [3; 18; 19; 20; 82]. Furthermore the supervised methodologies that can be used to learn the relation between visual features for the seen classes have to be adapted for the unseen domain, where the label is not available.

While some methods attempt to improve the projection in the transductive setup with pseudo-labeling [19; 83; 84; 85], quasi-fully supervised learning loss [86] or unsupervised-clustering [18], recent approaches take advantage of the generative architectures as in the inductive scenario [67; 77; 78; 87; 88]. [67] models the seen and unseen classes using an exponential family class-conditional distribution and using an expectation-maximization algorithm to alternate between inferring the labels and updating the parameter for the unseen classes. [87; 89] are GAN based, while [3; 77; 78] combine a VAE and a GAN architectures, all of them extending their architectures on the unseen domain by the addition of an unconditional discriminator. [88] uses a self-supervised learning algorithm combined with a Wasserstein GAN to better separate the seen and unseen domain.

# Chapter 3

## Decoupled Feature Generation

### 3.1 Context

As we presented in Chapter 1, the field of visual object recognition has seen a significant progress in recent years, mainly because of the availability of large-scale *annotated* datasets. However, annotating a big corpus of data for each of the categories to be recognized in a balanced way can be costly or is simply unfeasible due the well known long-tail distribution problem [9].

As a promising solution to the aforementioned issues, Zero-Shot Learning algorithms tackle the problem of recognizing novel categories [15; 16] by transferring auxiliary information from the seen classes to the unseen ones (see Chapter 2).

When transferring from the seen to the unseen classes, the main challenge is handling such category shift [3; 18; 19; 20]: in this Chapter, we evaluate this in the *Generalized Zero-Shot Learning* setup in which a method is evaluated on both seen and unseen classes, requiring to learn the latter without forgetting the former ones. In fact, given the separation of the data into labelled seen and unlabelled unseen instances, supervised training can be done for seen classes only, resulting in an unbalanced performance in Generalized Zero-Shot Learning. To address

this issue, various recent works proposed to augment the unseen-class data with synthetic labeled data [2; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92].

In this Chapter, we address the transductive Generalized Zero-Shot Learning problem, by introducing a novel, more effective, feature synthesis method able to balance the training process. In details, our approach builds upon the possibility of mimicking the human brain in hallucinating a mental imagery of a certain unknown category while reading a textual description of it. As pursued by a number of recent works [2; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92], conditional feature generation is adopted for this purpose. Specifically, images from the seen classes are fed into a backbone ResNet-101 network which pre-computes a set of real visual features [17]. Subsequently, through a Generative Adversarial Network (GAN) [36; 44; 45], the following min-max optimization game is solved: a generator network is asked to synthesize visual embeddings which should look real to a discriminator module. Since the generator network is conditionally dependent upon semantic embeddings, the trained model can be exploited to create synthetic features of the classes for which we lack visual labeled data. Afterwards, Generalized Zero-Shot Learning can be solved as simple classification problems through a softmax classifier trained on top of real features (from seen classes) and generated features (from the unseen ones).

In the literature, several variants have been attempted to improve the pipeline under the architectural point of view with the aim to solving efficiently Generalized Zero-Shot Learning: using an attribute regression module [91], easing the generator with a variational auto-encoder [59; 74; 77; 78; 90], adopting cycle-consistency [68], designing intermediate latent embeddings [72] or employing features-to-feature translation methods [2; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92]. On the one hand, as commonly done in adversarial training, we need to generate synthetic descriptors indistinguishable from a pool of pre-trained real features

used as reference. On the other hand, synthetic features are required to translate the semantic information into visual patterns, which are discriminative for the seen and unseen classes to be recognized.

We posit that resolving these two tasks within a single architecture is arguably difficult and we claim that this is the major limitation affecting the performance of the currently available feature generating schemes for Generalized Zero-Shot Learning. In fact, since adopting a single architecture for two tasks, one of them may be suboptimally solved with respect to the other, resulting in a poor modelling of either the visual or the semantic space.

Hence, differently to prior work, in this Chapter we propose to separately solve the two tasks, *decoupling* the feature generation stage to better tackle transductive Generalized Zero-Shot Learning. First, we train an unconditional generative adversarial network with the purpose of synthesizing features visually similar to the real ones in order to properly model their distribution in an unsupervised manner. Since our generation is not conditioned on semantic embeddings, we are sure to specifically model the visual appearance of our feature representations. Second, we encapsulate such visual information into a *structured prior*, which is used in tandem with a conditioning factor (here, the semantic embedding) to (conditionally) generate synthetic feature vectors. Because of the improved source of noise, we expect to enhance the semantic-to-visual translation as well, yielding visual descriptors with richer semantic content. The resulting architecture for decoupled feature generation is named DecGAN.

Since our DecGAN is decoupled into one unconditional and one conditional GAN-like branches, it is capable of exploiting the unlabeled visual data which are available in transductive Generalized Zero-Shot Learning. In fact, while we can compare the generated seen features with the real ones, both conditionally and unconditionally (since we have access to labels), we cannot do it for the un-

seen ones. Unseen classes are, in fact, not supported by annotated visual data, hence the conditional discriminator cannot “verify” them. We deem that our proposed architecture contributes in addressing this problem by *cross-connecting* the conditional branch with the unconditional one. In other words, we use the unconditional discriminator to evaluate the “quality” of the conditionally generated features. In this way, we decouple the feature generation, not only for the seen categories, but also for the unseen ones, resulting in a better modelling for both and improving the Generalized Zero-Shot Learning performance.

In summary, this work provides the following original contributions.

- We introduce the idea of *decoupling* feature generation for transductive zero-shot learning by encapsulating visual patterns into a structured prior, which is subsequently adopted to boost the semantically conditioned synthesis of visual features.
- We implement our idea through a novel architecture, termed DecGAN, which combines an unconditional and a conditional feature generation module, introducing a novel *cross-connected branch* mechanism able to decoupling feature generation for both seen and unseen categories.
- Through an extensive ablation study, we analyze each single component of our architecture. As compared to the transductive Generalized Zero-Shot Learning state-of-the-art, DecGAN outperforms it on CUB [6] and SUN [8] datasets (see Table 3.4, Section 3.4).

The rest of the Chapter is organised as follows. In Section 3.2, we outline the most relevant related work. In Section 3.3, we present the proposed approach, which is then experimentally validated in Section 3.4. The final conclusions are drawn in Section 3.5.

## 3.2 Related Work

In this Section, we will cover the most relevant related work in the field of transductive zero-shot learning. For more general details on other zero-shot paradigms, the reader can refer to Chapter 2.

Classical approaches in Generalized Zero-Shot Learning aim at learning a compatibility function between visual features and class embeddings, projecting them in a common space [17].

In order to exploit unlabeled data for the unseen classes in the transductive Generalized Zero-Shot Learning, [80] proposes a procedure to perform label propagation [81] as to simultaneously learn a representation for both seen and unseen classes. As shown in [17], this label propagation procedure can be extended to all the methods based on compatibility functions that map the visual features into the class embedding space [53]. Due to the shift between seen and unseen classes, the projection may struggle when switching from the seen to the unseen domain. To this end, in [18; 84; 85], the projection function is improved, while [19] tries to alleviate the issue using an ensemble of classifiers. Differently, we perform synthetic feature generation to produce labeled visual features for the unseen classes.

Generative approaches for transductive Generalized Zero-Shot Learning have been recently proposed [67; 77; 78; 87]. Using the taxonomy of generative models [93], the method proposed in [67] can be categorized as an explicit density model with tractable density. In fact, here it is proposed to model the density function of the conditional probability of the visual features given the class embedding through an exponential family of distributions. Among the profitable benefits of tractable density function, the computational pipeline becomes simpler and more efficient. However, constraining the density function limits the possibility to capture all the complexity of the data. Instead, our framework is based on

GANs [36; 44; 45], a direct implicit density model [93], and therefore, we do not impose any density function for the distribution from which we want to generate the visual features, but we let the model to directly learn it from the data.

In [87], a constraint is introduced in GAN training to improve the discriminative properties of the generated features. Specifically, a compatibility function  $f$  between visual features and class embeddings is learned, then the correlation between gradients of real and generated features in respect to  $f$  is maximized during GAN training.

In [78] and [77], a mixture of explicit and implicit models, a Variational Autoencoder (VAE) [46] and a GAN, is proposed. Specifically, a single generator/decoder is conditioned on the attribute embeddings and used to approximate the numerically intractable distribution of the visual features. By directly minimizing the divergence between the real visual features and the generated ones, the model learns 1) how to extract visual features for those classes which are not seen during training but only described through their attributes, and 2) how to mimic the distribution of visual features (with the addition of one adversarial categorization network in [77]).

Differently to [77; 78; 87], in our work, we propose a decoupled feature generation framework. Hence, instead of training one single conditional generator, we train an unconditional generator to solely capture the complexity of the visual data distribution, and we subsequently pair it with a conditional generator devoted to enrich the prior knowledge of the data distribution with the semantic content of the class embeddings.

To generalize the generative deep networks on the (unlabeled) unseen domain: [87] use an unconditional discriminator for both, seen and unseen data, and implicitly learns the class label information through the compatibility function; [77] apply a pseudo-labeling strategy; [78] use an additional unconditional



discriminator for the unseen data. Differently, we cross-connect our conditional and unconditional branches.

## 3.3 Decoupled Feature Generation

### 3.3.1 Notation and Problem Definition

We consider two disjoint sets of classes: the seen classes  $\mathcal{Y}^s$  and the unseen ones  $\mathcal{Y}^u$ , such that  $\mathcal{Y}^s \cap \mathcal{Y}^u = \emptyset$ . For the seen classes, a dataset of triplets  $(\mathbf{x}_s, y_s, c(y_s))$  is available:  $\mathbf{x}_s \in \mathcal{X}$  is the visual feature vector,  $y_s \in \mathcal{Y}^s$  is its class label and  $c(y_s)$  is the corresponding class embedding. Differently, for the unseen classes, in transductive Generalized Zero-Shot Learning we only have unlabeled visual features  $\mathbf{x}_u$ . The sets of the labels  $y_u$  of the unseen classes are described in terms of their semantic embeddings  $c(y_u)$ , as for the seen ones.

Given a test visual feature  $\mathbf{x}$ , the goal is to predict the corresponding class label  $y$  which can either belong to the seen or to the unseen classes.

For feature generation approaches, a conditional generator  $G$  is fed with random noise  $\mathbf{z}$ , and a class embedding  $c(y)$  and it synthesizes a feature vector which will be denoted by  $\tilde{\mathbf{x}}$ . Once  $G$  is trained, synthetic features  $\tilde{\mathbf{x}}_u$  are generated for the unseen classes and are used, together with  $\mathbf{x}_s$ , to train a softmax classifier which is responsible for the final recognition task.

### 3.3.2 Our Proposed Architecture: DecGAN

Looking at Figure 3.1, our proposed DecGAN architecture is composed of two cross-connected branches, which consist of two GANs - one unconditional (in yellow in the figure) and one conditional (in light blue), which are cross-connected forming a third cross-branch (in violet).

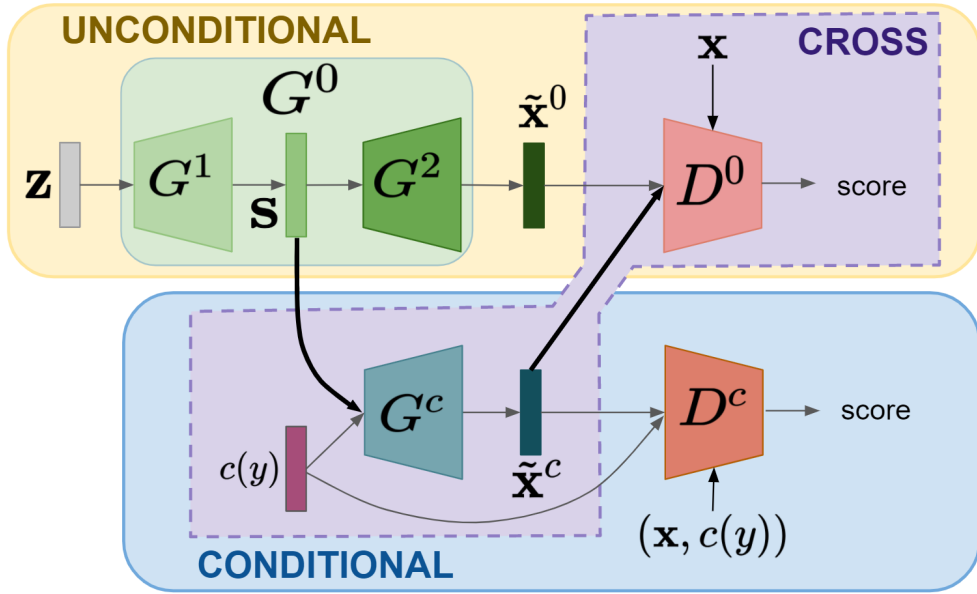


Figure 3.1: Our proposed DecGAN architecture is composed of two cross-connected branches consisting of two GANs: unconditional and conditional. The unconditional branch (yellow) is composed of generator  $G^0$  and discriminator  $D^0$ , and the conditional branch (light blue) is composed of generator  $G^c$  and discriminator  $D^c$ . An additional cross-branch (violet) is composed of  $G^c$  and  $D^0$ . Generator  $G^0$  is decomposed into  $G^1$  and  $G^2$ , such that given some random noise  $\mathbf{z}$ ,  $G^2(G^1(\mathbf{z})) = G^0(\mathbf{z}) = \tilde{\mathbf{x}}^0$ . The structured prior  $\mathbf{s} = G^1(\mathbf{z})$  is fed as input into  $G^c$  together with the class embeddings  $c(y)$ , for the sake of conditionally-generating visual descriptors  $\tilde{\mathbf{x}}^c$ . Best viewed in colors.

### 3.3 Decoupled Feature Generation

---

It can be noticed that there are two main ingredients: a *structured prior* and a *cross-connection* between the conditioned and unconditioned branches. Since the unconditional branch learns how to mimic the feature representation, regardless of the semantic class embeddings, this allows to generate a structured prior which can be shared across classes and adopted by the conditional branch to better perform the semantic-to-visual mapping. The cross-connection is fundamental as well: once synthetic features are conditionally generated, they can be checked to be realistic from the conditional generator only if they belong to the seen classes - for which we have labels. But, with the additional usage of the unconditional generator, we can also verify if the synthetic features from unseen classes are similar to real ones in distribution. This framework fully exploits the possibility of a transductive zero-shot learning.

The reader can refer to Figure 3.1 for a visualization and to the next paragraphs for the details on the design of our architecture.

#### 3.3.2.1 Unconditional Branch.

The unconditional branch is composed of the generator  $G^0$  and the discriminator  $D^0$ . The generator  $G^0$  is decomposed into  $G^1$  and  $G^2$ , such that, given some random noise  $\mathbf{z}$ ,  $G^2(G^1(\mathbf{z})) = G^0(\mathbf{z})$ . We refer to the output of  $G^1$  as the *structured prior*  $\mathbf{s}$ , that is  $\mathbf{s} = G^1(\mathbf{z})$ . The concatenation of  $\mathbf{s}$  and the class embeddings  $c(y)$  is passed as input to the conditional branch (see next paragraph). The unconditional branch is dedicated to learning the data distribution and we model it as a Wasserstein GAN (WGAN) [44; 45]. Hence, optimization is performed by minimizing the Wasserstein distance between the real data distribution and the synthetic's one by playing the following two-player game between  $G^0$  and  $D^0$  [44]:

$$\min_{G^0} \max_{D^0} \mathbb{E}_{\mathbf{x}}[D^0(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}^0}[D^0(\tilde{\mathbf{x}}^0)], \quad (3.1)$$

### 3.3 Decoupled Feature Generation

---

where  $\tilde{\mathbf{x}}^0 = G^0(\mathbf{z})$  denotes the features generated from the unconditional generator  $G^0$ . To regularize the min-max optimization, we use the following penalty term [45]:

$$\mathcal{R} = \mathbb{E}_{\hat{\mathbf{x}}}[(\|\nabla D^0(\hat{\mathbf{x}})\|_2 - 1)^2], \quad (3.2)$$

where  $\hat{\mathbf{x}} = \alpha \mathbf{x} + (1 - \alpha)\tilde{\mathbf{x}}^0$  with  $\alpha \sim \mathcal{U}(0, 1)$ .

#### 3.3.2.2 Conditional Branch.

To learn how to translate the semantic content of the class embeddings  $c(y)$  we model the conditional branch with the extension of the WGAN to a conditional model [2]. The conditional branch is composed of the generator  $G^c$  and the discriminator  $D^c$ . In this architecture, both the generator and the discriminator are conditioned on the class embeddings. The generator  $G^c$  takes as input the structured prior  $\mathbf{s}$  and it is conditioned on the class embeddings, learning how to enrich the information about the data distribution contained in  $\mathbf{s}$  with the semantic content of  $c(y)$ . The generated features  $\tilde{\mathbf{x}}^c = G^c(\mathbf{s}, c(y))$  are then evaluated by the discriminator  $D^c$  together with the class embedding that generated them, and compared to real data pairs  $(\mathbf{x}, c(y))$ .

With this architecture,  $G^c$  learns how to enrich  $\mathbf{s}$  with the content of the class embeddings. The quality of the relation between the generated visual features and the semantic content is then evaluated by  $D^c$ . The optimization is carried out through

$$\min_{G^c} \max_{D^c} \mathbb{E}_{\mathbf{x}}[D^c(\mathbf{x}, c(y))] - \mathbb{E}_{\tilde{\mathbf{x}}^c}[D^c(\tilde{\mathbf{x}}^c, c(y))], \quad (3.3)$$

with the regularization term [2]:

$$\mathcal{R} = \mathbb{E}_{\hat{\mathbf{x}}}[(\|\nabla D^c(\hat{\mathbf{x}}, c(y))\|_2 - 1)^2], \quad (3.4)$$

where  $\hat{\mathbf{x}} = \alpha \mathbf{x} + (1 - \alpha)\tilde{\mathbf{x}}^c$  with  $\alpha \sim \mathcal{U}(0, 1)$ .

### 3.3 Decoupled Feature Generation

---

We also add the regularization term introduced by [68]. That is, given a pre-trained linear module  $A$  and  $\tilde{a} = A(\hat{\mathbf{x}})$  the reconstruction of  $c(y)$  given  $\hat{\mathbf{x}}$ , we add the reconstruction loss:

$$\mathcal{R}_{rec} = \|c(y) - \tilde{a}\|_2^2. \quad (3.5)$$

#### 3.3.2.3 Cross Branch.

Because labeled data are not available for the unseen classes, we cannot feed the conditional discriminator  $D^c$  with them. To exploit the unlabeled data we propose to conditionally generate the visual features  $\tilde{\mathbf{x}}^c$  and evaluate them only by their distribution using  $D^0$ . Thus in this setting we do not condition both the generator and the discriminator, as is commonly done in GAN based conditional generation, but we only condition the generator. Hence, optimization is obtained by

$$\min_{G^c} \max_{D^0} \mathbb{E}_{\mathbf{x}}[D^0(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}^c}[D^0(\tilde{\mathbf{x}}^c)], \quad (3.6)$$

adapting as consequence the regularization term on the gradients as

$$\mathcal{R} = \mathbb{E}_{\tilde{\mathbf{x}}}[(\|\nabla D^0(\hat{\mathbf{x}})\|_2 - 1)^2], \quad (3.7)$$

where  $\hat{\mathbf{x}} = \alpha\mathbf{x} + (1 - \alpha)\tilde{\mathbf{x}}^c$  with  $\alpha \sim \mathcal{U}(0, 1)$  and adding the reconstruction loss defined in equation (3.5).

**The origin of the proposed architecture.** Our work is inspired by FusedGAN [94], which combines two GANs to improve image generation in a semi-supervised setup. Differently, in our case, we handle feature generation in the zero-shot case, so we have no annotated data at all for some of the classes and we need to generate them. To solve this problem, differently from FusedGAN, we cross-connect the two branches to transfer the knowledge of the seen domain to the unseen one.

### 3.3 Decoupled Feature Generation

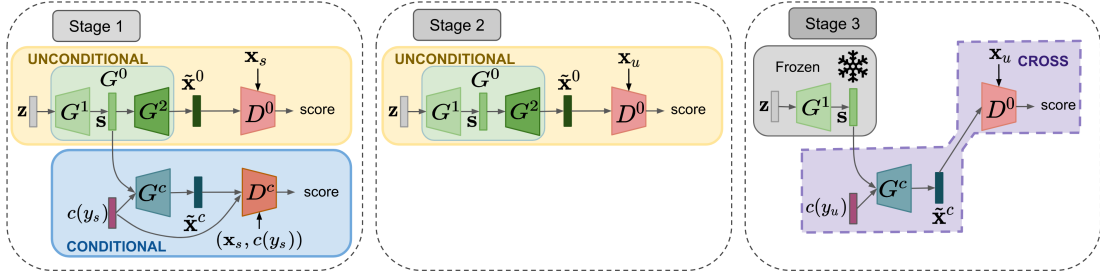


Figure 3.2: DecGAN training is performed in 3 stages. Stage 1 performs an alternate training on the conditional and the unconditional branch using seen data. Stage 2 uses the unconditional branch to fine-tune  $G^0$  using unseen data to improve the structured prior  $\bar{s}$ . Finally, Stage 3 carries out the fine-tuning of  $G^c$ , feeding the cross-branch with unseen data.

#### 3.3.3 Training Methodology

To train the proposed DecGAN, we propose a three-staged training strategy, which is explained beneath and sketched in Figure 3.2.

1. In the first stage, we optimize both the conditional and the unconditional branch using only data from the seen classes. We seek to achieve decoupled feature generation for the seen categories in a way that, while  $G^0$  learns how to model the data distribution,  $G^c$  learns how to enrich the structured prior with the content of the class embeddings. We perform an alternate training strategy in which, first, we update  $D^0$  and  $G^0$  using equations (3.1) and (3.2). Then, we update  $D^c$  and  $G^c$  using equations (3.3) and (3.4). A full update step consists of  $k$  updates of  $D^0$ , 1 update of  $G^0$ ,  $k$  updates of  $D^c$  and 1 update of  $G^c$  in sequence with  $k > 1$  [45]. Here, we chose  $k = 5$  as done in [2].
2. In the second training stage, we want to take advantage of the unseen unlabeled data to add into the structured prior the information of the unseen data distribution. To reach our goal, we use the unseen data to fine-tune  $D^0$  and  $G^0$  using equations (3.1) and (3.2).

3. The third stage consists in the fine-tuning of the conditional generator  $G^c$  on the unseen data. Using structured prior, generalized over the unseen classes in the previous stage, we condition  $G^c$  with the embeddings of the unseen classes to reinforce its ability to translate semantic content into visual features in the unseen domain. That is, we use equations (3.6) and (3.7) to update  $D^0$  and  $G^c$  using unseen data.

### 3.3.4 Implementation Details

We implement  $G^2$  and  $G^c$  as single hidden layer neural networks with hidden layer of size 4096 and leaky ReLU activation and output layer that has the size of the visual feature vectors, 2048, and ReLU activation. In  $G^1$ , a leaky ReLU is used as the activation function (without hidden layer). Specifically,  $G^0$  is a 2-hidden layer neural network and we use its first layer as structured prior. The size of the structured prior (the output of  $G^1$ ) is fixed to 1024. The size of noise  $\mathbf{z}$  is fixed to 512 and is sampled from a multivariate normal distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ , where  $\mathbf{0}$  is the 512-dimensional vector of zeros and  $\mathbf{I}$  is the 512-dimensional identity matrix.  $D^0$  and  $D^c$  are neural networks composed of a single hidden layer of size 4096 (with leaky ReLU activation) and by an unconstrained real number as output.

## 3.4 Experiments

### 3.4.1 Datasets and Benchmarks

We evaluate proposed DecGAN on standard benchmark datasets for Generalized Zero-Shot Learning. We considered Oxford Flowers (FLO) [7], SUN Attribute (SUN) [8] and Caltech-UCSD-Birds 200-2011 (CUB) [6]. FLO consists of 8,189 images of 102 different types of flowers, SUN 14,340 images of scenes from 717

Dataset	att	stc	$\#\mathcal{Y}^s$	$\#\mathcal{Y}^u$
FLO [7]	-	1024	62+20	20
SUN [8]	102	-	580+65	72
CUB [6]	312	-	100+50	50

Table 3.1: Statistics of considered datasets: number of seen classes  $\#\mathcal{Y}^s$  (training + validation), unseen classes  $\#\mathcal{Y}^u$ , dimension of attribute (att) annotation and sentences (stc) extracted features.

classes and CUB of 11,788 images of 200 different types of birds.

For SUN and CUB, we use manually annotated attributes [17]. Because for FLO the attributes are not available, we follow [78] in using 1024-dim sentence embedding extracted by the character-based CNN-RNN [25] from fine-grained visual descriptions of the images. Statistics of the datasets are available in Table 3.1. For a fair comparison, we split the classes of SUN and CUB between seen and unseen using the splits proposed by [17]. For FLO, we use the splits as in [25]. For all datasets, the visual features are chosen as the 2048-dim top-layer pooling units of the ResNet-101 [33], provided by [17].

As evaluation metrics, the Generalized Zero-Shot Learning setup, we measure the performance as harmonic mean between seen and unseen accuracy, each one computed as top-1 classification accuracy on seen and unseen classes [17].

### 3.4.2 Ablation Study

In this Section, we will perform an accurate ablation analysis on the different components of our proposed DecGAN architecture. Specifically, we will separately evaluate the impact on performance of each of the three branches which endows DecGAN: namely, the unconditioned, conditioned and cross-branch as presented in Section 3.3.2. We will also pay attention to the effects of the different stages of our training pipeline: the first stage is evaluated in stand-alone fashion, while we also provide experimental evidence for the effect of removing the second stage.



### 3.4 Experiments

	FLO			SUN			CUB		
	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$
DecGAN <sup>(Stg1)</sup>	58.1	79.8	67.2	45.0	34.5	39.1	44.1	56.7	49.8
DecGAN <sup>(Stg3)</sup>	4.7	82.2	8.9	1.2	30.1	2.3	2.0	35.3	3.8
DecGAN <sup>(-Stg1)</sup>	5.8	73.9	10.1	1.3	39.1	2.5	1.7	35.3	3.2
DecGAN <sup>(-Stg2)</sup>	71.1	50.5	80.1	53.2	44.2	48.2	55.1	66.6	60.3
DecGAN <sup>(-Stg3)</sup>	50.5	80.1	62.0	44.5	34.7	38.3	45.3	53.8	49.1
DecGAN	73.0	92.2	<b>81.5</b>	57.2	44.3	<b>49.9</b>	59.1	68.4	<b>63.4</b>

Table 3.2: We assess the impact on performance of the presence/absence of each stage of the training pipeline of our DecGAN model. We report top-1 accuracy on seen classes  $\mathbf{a}_s$  and unseen classes  $\mathbf{a}_u$  and their harmonic mean  $\mathbf{H}$ . Best  $\mathbf{H}$  values are highlighted in bold. All results are reported by averaging accuracies over 5 different runs.

#### 3.4.2.1 The Impact on Performance of each Training Stage.

To better analyse our composite training pipeline, in Table 3.2, we present an ablation study to assess the impact on performance of each of the three stages of our training pipeline. Precisely, we evaluate the drop in performance resulting from removing any of the aforementioned stages from the full training pipeline of DecGAN: when either removing the first, second or third stage, we obtain DecGAN<sup>(-Stg1)</sup>, DecGAN<sup>(-Stg2)</sup> and DecGAN<sup>(-Stg3)</sup>, respectively. We also assess the performance of the first and third stage separately (DecGAN<sup>(Stg1)</sup> and DecGAN<sup>(Stg3)</sup>) Note that, we cannot evaluate the stage 2 in a standalone fashion since such stage lacks of a conditional feature generation pipeline from which we can sample features for the seen/unseen classes (see Figure 3.2).

With respect to the performance of the full DecGAN model, the first stage always achieve a suboptimal performance, and this can be clearly explained by the fact that, DecGAN<sup>(Stg1)</sup> exploits data from the seen classes only (inductive setup). The poor performance of DecGAN<sup>(Stg3)</sup> reveals that without the conditional discriminator, which takes as input the visual features and the related class embeddings (consequently learning the relation between visual features and class

### 3.4 Experiments

	FLO			SUN			CUB		
	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$
Not decoupled	69.5	91.4	79.0	52.7	44.3	48.1	54.3	66.7	59.9
Decoupled	73.0	92.2	<b>81.5</b>	57.2	44.3	<b>49.9</b>	59.1	68.4	<b>63.4</b>

Table 3.3: The effect of decoupling the feature generation stage. We compare the decoupled approach of DecGAN to the not decoupled baseline. We report top-1 accuracy on seen classes  $\mathbf{a}_s$  and unseen classes  $\mathbf{a}_u$  and their harmonic mean  $\mathbf{H}$ . Best  $\mathbf{H}$  values are highlighted in bold. All results are reported by averaging accuracies over 5 different runs.

embeddings), the generator does not learn to generate class dependent visual features. When removing the first stage, the performance is comparable with DecGAN<sup>(Stg3)</sup>: this shows how pivotal this first step is for the whole pipeline. During the second stage, we fine-tune the structured prior adding information about the data distribution of the unseen visual features, but we update the conditional generator on the new structured prior only in Stage 3. The misalignment between the structured prior that the conditional generator expects as input and the updated one it receives leads in a performance drop when comparing DecGAN<sup>(Stg1)</sup> and DecGAN<sup>(-Stg3)</sup>. However, the importance of the transitory Stage 2, is highlighted by the difference in performance between DecGAN and DecGAN<sup>(-Stg2)</sup>. The third stage has a clear effect on performance: fine-tuning the conditional generator over the unseen data always boosts the performance, no matter if the second stage was performed or not.

#### 3.4.2.2 The Effect of the Decoupled Feature Generation.

We posit that the main advantage of DecGAN is the possibility of decoupling the feature generation stage, as to tackle two problems separately: 1) the generation of features which are visually similar to the real ones, and 2) the translation of semantic patterns from attributes to features. We want now to prove that the aforementioned separation of the tasks leads to a superior performance if

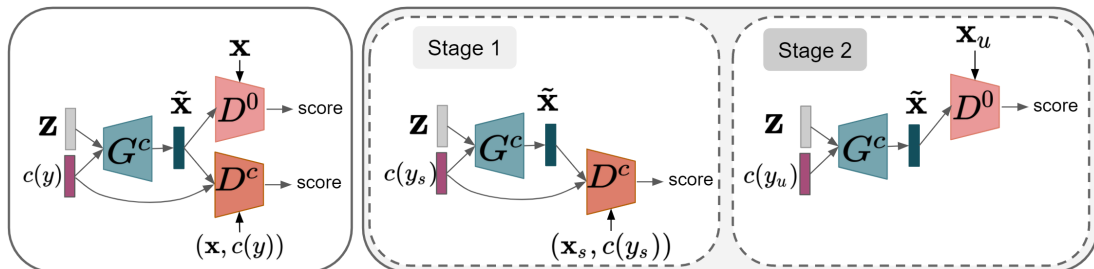


Figure 3.3: Baseline architecture. *Left*: a visualization of the architectural design of the baseline. *Right*: similarly to our proposed DecGAN, we adopt a staged training to optimize the baseline.

compared to a model which tries to perform both tasks jointly. To do so, we compare the proposed decoupled feature generation (achieved through DecGAN) with a baseline model in which we perform a similar staged training, without performing decoupling.

Specifically, we consider the architecture represented in Figure 3.3, which is described beneath. It is composed of a single conditional generator  $G^c$  and two discriminators, one conditional  $D^c$  and one unconditional  $D^0$ , implemented through Wasserstein GANs subjects to gradient penalty loss and reconstruction loss in the same way of our DecGAN. Moreover, similarly to our DecGAN, we train this architecture in two stages (see Figure 3.3). In the first stage, we train  $G^c$  together with  $D^c$  using only seen data, and in the second stage, we train  $G^c$  together with  $D^0$  using unseen data. Differently to DecGAN, generation is performed with a single generator that has to learn both, the data distribution of the real visual features and how to translate the semantic content of the class embeddings into them, without taking advantage of the structured priors and the decoupled features generation. As for DecGAN,  $G^c$  and  $D^0$  and  $D^c$  for this baseline are one-hidden layer neural networks with hidden layer of size 4096 with leaky ReLU activation. The size of the noise is fixed to 1024, the same of the structured prior  $\mathbf{s}$ .

The results of this analysis are presented in Table 3.3 and discussed in the following.

The effect of decoupling is clearly visible on all the 3 benchmarks showing that it is always advantageous since leading to a superior performance, when considering all error metrics adopted in this Chapter: accuracy over unseen and seen classes  $\mathbf{a}_u$ ,  $\mathbf{a}_s$  and harmonic mean  $\mathbf{H}$ . When we do not perform decoupling, the resulting performance is comparable to the one achieved by DecGAN<sup>(-Stg2)</sup> (check Tables 3.2 and 3.3). We think that this is an effect of what happens to the structure prior, which is first trained on the seen data and the conditional generator, and then fine-tuned on the unseen ones. Such discontinuous usage of seen and unseen data leads the generator to use the structured prior as a random input, since it is not able to read the visual information which is encapsulated inside.

### 3.4.3 Comparison with the State-of-the-Art in transductive Generalized Zero-Shot Learning

In this Section, we report the key benchmark against the state-of-the-art transductive Generalized Zero-Shot Learning. The methods are the projection with visual structure constrain (CDVSc) [18], the effective deep embedding (EDE<sub>ex</sub>) [84], the progressive ensemble network (PREN) [19], the domain-invariant projection (Full DIPL) [85], the attribute-based latent embedding (ALE) [53], the generative framework based on a family of Gaussian distributions (GFZSL) [67], the discriminative semantic representation learning (DSRL) based on a non-negative matrix factorization approach [80], the feature generation approach based on a paired network and variational auto-encoder GAN and VAE [77; 78] (f-VAEGAN-D2 and Z-VAE-GAN) and the addition of the gradient matching loss during the gan training [87] (GMN). We additionally report some classical inductive meth-

ods as [53; 55; 58; 62], as well some generative inductive methods based on GAN [2; 68], VAE [59; 90] or combination of them [78]. We selected the three publicly available benchmark datasets (FLO, SUN and CUB) presented in Section 3.4.1. We present the results through mean over five different runs at a fixed number of DecGAN training epochs. We report results obtained in Table 3.4.

In Table 3.4, we show how our proposed decoupled feature generation, implemented through our DecGAN model, is capable of improving in performance prior methods.

Among the inductive zero-shot learning methods, DecGAN sets up sharp improvements in performance: methods such as CADA-VAE [59] are improved in the scored harmonic mean  $\mathbf{H}$  by +9.3% on SUN and by +11.0% on CUB. Similarly, DecGAN is capable of outperforming cycle-WGAN [68] on FLO (+16.3%), SUN (+10.5%) and CUB (+10.4%). Actually, even the performance of DecGAN in the first training stage is superior to cycle consistency: DecGAN<sup>(Stg1)</sup> improves cycle-WGAN by +2.0% of FLO.

A solid performance is shown also when benchmarking the state-of-the-art in the transductive generalized zero-shot learning setup. On FLO, DecGAN improves several prior methods by margin, in terms of  $\mathbf{H}$ : +59.3% with respect to ALE [17], +47.7% with respect to GFZSL [67] and +43.6% with respect to DSRL [80]. The only method reported in Table 3.4 which is slightly superior to us is f-VAEGAN-D2 [78] and the reason for that is the usage of two feature generation schemes: a variational autoencoder and a GAN. We therefore deem that our idea is still competitive: by means of our structured prior, we can almost match the performance of a method which uses twice the number of feature generators. This evidently shows DecGAN as a balance between model light-weighting and performance. On the other two benchmark datasets, DecGAN improves f-VAEGAN-D2 [78] by +0.3% on SUN and +0.2% on CUB and, similarly, surpasses in perfor-

### 3.4 Experiments

Table 3.4: Results in Generalized Zero-Shot Learning. We report top-1 accuracy on seen classes  $\mathbf{a}_s$  and unseen classes  $\mathbf{a}_u$  and their harmonic mean  $\mathbf{H}$ . First and second best values are highlighted in bold and italic, respectively, for  $\mathbf{H}$ . Inductive (I) and Transductive (T) methods are reported. †: results not reported because of the usage of different class embeddings, which are not comparable. Our results are presented by averaging performance scores over 5 different runs.

		FLO			SUN			CUB		
		$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$	$\mathbf{a}_u$	$\mathbf{a}_s$	$\mathbf{H}$
ESZSL [55]		-	-	-	11.0	27.9	15.8	12.6	63.8	21.0
ALE [53]		-	-	-	21.8	33.1	26.3	23.7	62.8	34.4
SynC [62]		-	-	-	7.9	43.3	13.4	11.5	70.9	19.8
LATEM [3]		-	-	-	14.7	28.8	19.5	15.2	57.3	24.0
f-CLSWGAN [2]	I	59.0	73.8	65.6	42.6	36.6	39.4	43.7	57.7	49.7
f-VAEGAN [78]		56.8	74.9	64.6	38.0	45.1	41.3	48.4	60.1	53.6
cycle-WGAN [68]		61.6	69.2	65.2	33.8	47.2	39.4	47.9	59.3	53.0
SyntE [90]		-	-	-	40.9	30.5	34.9	41.5	53.3	46.7
CADA-VAE [59]		-	-	-	47.2	35.7	40.6	51.6	53.5	52.4
DSRL [80]		26.9	64.3	37.9	17.7	25.0	20.7	17.3	39.0	24.0
GMN [87]		-	-	-	57.1	40.7	47.5	†	†	†
CDVSc [18]		-	-	-	27.8	63.2	38.6	37.0	84.6	51.4
PREN [19]		-	-	-	35.4	27.2	30.8	35.2	55.8	43.1
Z-VAE-GAN [77]		-	-	-	53.1	35.8	42.8	64.1	57.9	60.8
ALE <sub>trans</sub> [17]	T	13.6	61.4	22.2	19.9	22.6	21.2	23.5	45.1	30.9
Full DIPL [85]		-	-	-	-	-	-	41.7	44.8	43.2
EDE <sub>ex</sub> [84]		-	-	-	47.2	38.5	42.4	54.0	62.9	58.1
GFZSL [67]		21.8	75.8	33.8	0.0	41.6	0.0	24.9	45.8	32.2
f-VAEGAN-D2 [78]		78.7	87.2	<b>82.7</b>	60.6	41.9	<i>49.6</i>	61.4	65.1	<i>63.2</i>
DecGAN (ours)		73.0	92.2	<i>81.5</i>	57.2	44.3	<b>49.9</b>	59.1	68.4	<b>63.4</b>

mance recent prior art such as PREN [19] (+19.1% on SUN and +20.3% on CUB) or Z-VAE-GAN [77] (+7.1% on SUN and +2.6% on CUB).

## 3.5 Conclusions

In this Chapter, we address a major limitation of the mainstream approach in (generalized) zero-shot learning, consisting in the necessity of solving two problems with a single computational pipeline: 1) capturing the distribution of visual features in order to generate realistic descriptors, and 2) translating semantic attributes into visual patterns. Therefore we proposed DecGAN, which decouples the aforementioned problems, by means of an unconditional GAN generating a structured prior. The latter can be used to improve the conditional generation of visual features. The overall architecture has a staged training, whose steps have been validated in a broad experimental comparison, assessing that this computational setup is particularly favorable for the transductive Generalized Zero-Shot Learning setup. In fact, DecGAN is improving in performance previous state-of-the-art on challenging public benchmark datasets.

# Chapter 4

## Open Zero-Shot Learning

### 4.1 Context

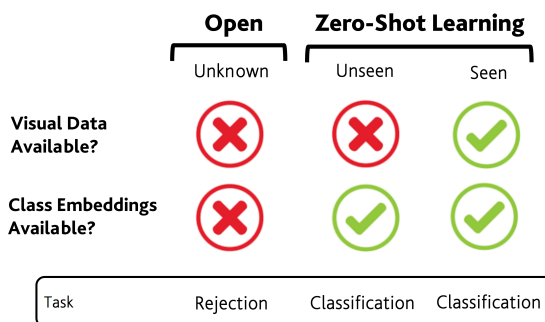


Figure 4.1: **Open Zero-Shot Learning**, a framework where we aim at classifying seen and unseen classes (for which no visual data of the latter is given) while also rejecting (*i.e.*, refusing to take any decision on) unknown classes. Neither visual data nor class embeddings are available for unknown classes.

As we presented in the previous Chapters, Zero-Shot Learning aims at recognizing novel classes by performing knowledge transfer taking advantage of the class embeddings. Since the initial publications [15; 16; 51], Zero-Shot Learning has attracted increasing attention and many competitive solutions to the problem have been proposed (see Section 2.5). However the initial setup was considered



a major limitation for real-world application and thus the more realistic and competitive Generalized-Zero Shot Learning have been proposed. Generalized Zero-Shot Learning considers both the unseen and the seen classes for evaluation, differently from the standard Zero-Shot Learning that only considers the seen classes.

In Generalized Zero-Shot Learning, the challenge is to overcome the bias of the model towards predicting the classes on which it has been directly trained on, and for which it is much more confident in forecasting. To solve the extreme imbalance of the Generalized Zero-Shot Learning framework, much effort has been exerted to perform synthetic feature augmentation for the unseen classes [2; 4; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92]. By exploiting deep generative models, as Generative Adversarial Networks (GANs) or Variational Auto-Encoders (VAEs), it is indeed possible to take advantage of the class embeddings to generate class consistent features for the unseen classes by training on the seen ones, leading to remarkable performances in Generalized Zero-Shot Learning.

However, we claim that the assumption of knowing in advance the full set of classes, the closed-set assumption, and their class embeddings is still a strong limitation for Generalized Zero-Shot Learning in real world applications. In fact, while it is reasonable to assume that we can describe all the seen classes with the class embeddings, it seems less reasonable not only to know, but also to describe with the rich semantic content of the class embeddings, all the classes for which we have no visual training data.

***We introduce a new paradigm, Open Zero-Shot Learning*** (Figure 4.1). *Open Zero-Shot learning* overcomes the closed-set assumption and goes to the open-set scenario by considering a possible infinite set of classes at inference time. As a consequence, we have three types of classes: 1) *the seen*, for which we have

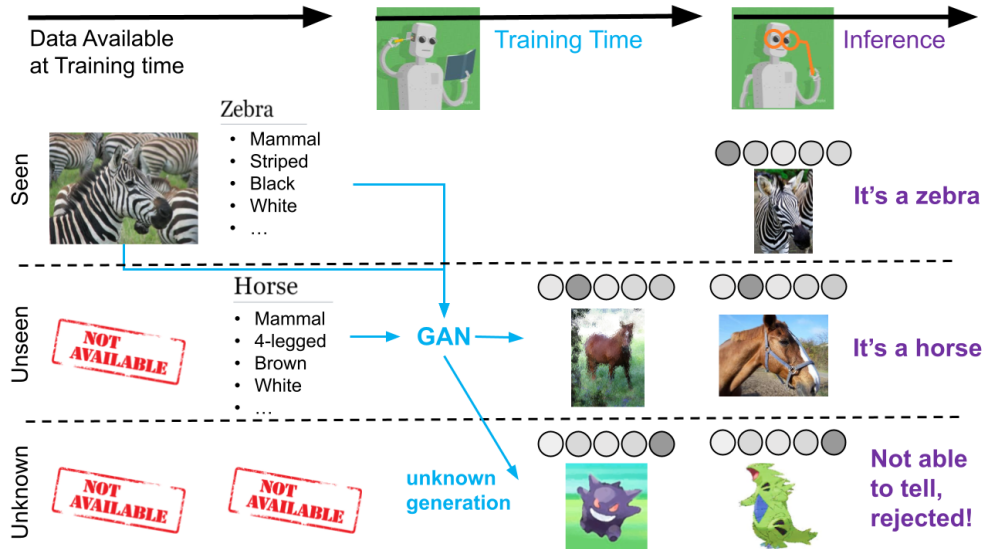


Figure 4.2: **The proposed pipeline for Open Zero-Shot Learning.** We synthesize visual descriptors from seen and unseen classes, using a Generative Adversarial Network (GAN). We also learn how to perform unknown generation and synthesize descriptors (represented by  $\circ\circ\circ\circ$ ), even for the unknown classes, and better precondition a classifier in classifying seen/unseen and reject unknown, with the usage of Openmax [1].

visual data and class semantic descriptors, 2) *the unseen*, for which we have only class embeddings, and 3) *the unknown*, for which we have neither the visual data nor the (semantic) class embeddings. Thus, Open Zero-Shot Learning extends Generalized Zero-Shot Learning with the possibility of performing recognition in the open-set regime [95] where inference has to be jointly performed over seen, unseen and unknown classes in order to classify seen and unseen, and reject unknown ones.

***We build Open Zero-Shot Learning as the open-set generalization of Generalized Zero-Shot Learning.*** To warm up the research community towards the solution of Open Zero-Shot Learning, we design evaluation protocols, extracting unknown classes as a subpart of unseen classes from typical Generalized Zero-Shot Learning benchmark datasets used in the related state of the art [2;

4; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92]. We will make these splits publicly available so as to ease the research community in this direction, and we also propose error metrics to allow fair and reproducible comparison across different algorithmic solutions tackling Open Zero-Shot Learning. We also extend prior Generalized Zero-Shot Learning error metrics (harmonic mean of the per-class average accuracy [17]) to better handle the open set scenario. In particular, we consider F1-score between seen and unseen average precision and/or recall scores to better account for successful rejections.

***We approach Open Zero-Shot Learning by synthesizing the unknown.***

(Figure 4.2). In Generalized Zero-Shot Learning, GANs or alternative generative methods [2; 4; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92]) generate visual features conditioned on class embeddings in order to synthesize descriptors for the unseen classes and train a softmax classifier on top of them as well as of real seen features. We purport that we can easily extend this state-of-the-art paradigm to Open Zero-Shot Learning by replacing the standard softmax classifier with Openmax<sup>1</sup> [1]. We provide a preliminary exploratory analysis, evaluating both baseline methods (*e.g.*, Generalized Zero-Shot Learning feature generator simply borrowed for Open Zero-Shot Learning) and our novel idea to synthesize unknown class embeddings and using them to generate unknown visual features, which we implemented through a variation of Wasserstein GANs [2; 3; 45], which we term VAcWGAN (variationally-conditioned Wasserstein GAN). VAcWGAN optimizes a conditional generative process on semantic embeddings (so that, we first “synthesize the unknown” and then we generate unknown visual features). Despite this approach being arguably harder (since we attempt to generate something we

---

<sup>1</sup>Openmax [1] augments the softmax classes’ bins (out of which probabilities are arg-maxed to compute predictions) by introducing an extra-bin estimating the probability to reject an instance. Thanks to Openmax, we can still cast recognition over seen & unseen classes, and rejection over unknown classes through a single arg max step.

do not see nor know), our experimental evidence shows some potential which we deem worthy to be further investigated by the computer vision community.

## 4.2 Related Work

**Generalized Zero-Shot Learning.** Feature generating networks are surely a “big thing” for Generalized Zero-Shot Learning [2; 4; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92]. As proposed by [78] and [96] almost independently, a (Wasserstein) GAN, conditioned on class embeddings, is paired with a classification loss in order to generate sufficiently discriminative CNN features, which are then fed to a softmax classifier for the final inference stage.

Recently, several modifications have been adopted to improve feature generation for ZSL, for instance, by either replacing the GAN with a variational autoencoder [74; 90] or using the latter two models in parallel [77; 78], cycle consistency loss [68; 91] or contrastive loss [4]. In [72], class embeddings are regressed from visual features, while semantic-to-visual generation is inverted with another generative, yet opposite, visual-to-semantic stream [59; 97].

Differently to all these methods, our GAN-based architecture is different in the way it synthesizes class embeddings for the unknown classes. Please note that two recent solutions applied a similar idea for the sake of learning a better noise for the GAN [92] [98], but, *to the best of our knowledge, we are the first to synthesize class embeddings*. As a concurrent work to ours, [99] seems to approach the open-set scenario as well: but, rather than building upon the “standard” (G)ZSL protocol used in computer vision [17], it approaches the “compositional setup”. That is, seen classes are defined as combinations of tags (*e.g.*, “wet dog” or “furry cat”) and inference has to be done on unknown combinations (*e.g.*, “furry dog”). Differently to [99], we put no prior on the classes we need to generalize onto

(unseen and unknown mainly) as we tackle the challenging generalization gap that requires us, for example, to reject unknown dolphins while not forgetting how to classify seen humpback whales and unseen blue whales.

**Rejecting Unknown Categories.** After the initial formalization of [95] on how to learn in the open set paradigm, many approaches have proposed for letting traditional machine learning models to deal with the *unknown* [100; 101; 102; 103; 104; 105; 106; 107; 108; 109; 110; 111; 112; 113; 114]. The interested reader may refer to [115] for an overview.

Leveraging the widespread usage of softmax classifier as the default classifier of deep neural networks, Openmax [1], proposed a meta-learning algorithm so that the probability of a data point to be an outlier can be modelled generating an extra-bin which estimate the probability of rejecting the given instance when recognized as outlier. Since then, a few algorithmic variants have been applied to Openmax, ranging from the usage of data-driven preconditioning [48] to counterfactual learning [116]. In our case, we do not change Openmax in its algorithmic implementation, but, rather, we fed it by data which are “much more difficult” to manage as compared to prior art. In fact, we ask Openmax not only to recognize seen classes, but also two different types of categories for which visual data are not available (unseen and unknown). Prior art in Openmax only considers seen vs. unknown [1] or seen vs. unseen [117] and, to the best of our knowledge, we are the first to jointly consider seen, unseen and unknown.

## 4.3 Problem Formulation

In this Section, we relax the closed-set assumption that constraints Generalized Zero-Shot Learning methods in knowing class embeddings for all categories (both seen  $\mathcal{S}$  and unseen ones  $\mathcal{U}$ ): we therefore attempt to reject unknown categories

while not forgetting seen and unseen ones. We do so by proposing Open Zero-Shot Learning, in which we augment  $\mathcal{S}$  and  $\mathcal{U}$  with a third set of classes, dubbed *unknown*, and denoted by  $\Omega$ . Unknown classes are deprived of both visual data and class embeddings (see Fig. 4.1). We formalize the Open Zero-Shot Learning problem by instantiating evaluation protocols, datasets and error metrics. We root these in Generalized Zero-Shot Learning to ease the transfer of the zero-shot learning community towards the new Open Zero-Shot Learning paradigm.

### 4.3.1 Open Zero-Shot Learning evaluation protocol.

In Generalized Zero-Shot Learning, seen classes  $\mathcal{S}$  are provided of data which are triplets  $[\mathbf{x}, y, \mathcal{C}_y]$ :  $\mathbf{x}$  are vectorial visual embeddings extracted from a deep convnet (usually, ResNet101 [17]) fed by related images,  $y$  is the class label and  $\mathcal{C}_y$  is a class embeddings (*e.g.*, a list of manually-defined attributes describing the class that are converted into float numbers ranged in  $[0, 1]$  through Osherson’s default probability scores [16]). Unseen classes  $\mathcal{U}$  are instead only given of class embeddings (and labels)  $[y, \mathcal{C}_y]$  at training time, hence totally missing visual data.

In Open Zero-Shot Learning, together with the recognition of seen and unseen classes, we encompass potentially infinitely many classes at inference time. In fact, in addition to classify examples from  $\mathcal{S}$  and  $\mathcal{U}$ , we also consider examples to be rejected since belonging to *unknown* categories we never observed before (no visual data available) and without class embeddings disclosed to the learner. Thus, unknown classes, denoted by  $\Omega$ , are totally deprived of any visual or semantic information.

Therefore, the task is to train a zero-shot learner to handle the open-set scenario where, not only it has to recognize any unobserved test instance for which visual patterns are apparently matching semantic information of class embeddings, but it has also to avoid to take any decision on instances that seem to

have a visual content that is not compatible with any prior semantic knowledge encapsulated in seen and unseen class embeddings.

### 4.3.2 OSZL datasets.

In order to allow practitioners to provide experimental results in both the closed-set, *i.e.*, Generalized Zero-Shot Learning, and the open-set, the proposed Open Zero-Shot Learning, we build Open Zero-Shot Learning benchmark datasets rearranging Generalized Zero-Shot Learning ones. Specifically, we consider Animals with Attributes (AWA) [5], Caltech-UCSD Birds 200 2011 (CUB) [6], Scene Understanding (SUN) [8], and Oxford Flowers 102 (FLO) [7] since they are, by far, ubiquitous in Generalized Zero-Shot Learning literature [2; 59; 68; 72; 77; 78; 87; 90; 91; 92]. We leverage the “Proposed Splits” [17] to be still enabled to use ImageNet pre-trained models to obtain visual descriptors (which are actually already pre-computed from a ResNet-101 and shared by the authors of [17]) and we stick to their proposed subdivision into seen and unseen classes. We select unknown categories by sampling from unseen classes. In short, we propose to sample 50% of the unseen classes of [17] and transform them to unknown classes, keeping the remaining 50% as unseen categories in Open Zero-Shot Learning. A complete list of seen, unseen and unknown classes for the selected four benchmark datasets is available in the Supplementary Material A.

### 4.3.3 Error metrics.

In Generalized Zero-Shot Learning, the performance is usually [17] evaluated using the harmonic mean

$$H_{\text{GZSL}} = \frac{2R_s R_u}{R_s + R_u}, \quad (4.1)$$

### 4.3 Problem Formulation

---

between each per-class accuracy  $R_s$  and  $R_u$ , computed over seen and unseen classes, respectively.  $R_s$  and  $R_u$  are defined as:

$$R_s = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} R_s = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{TP_s}{TP_s + FN_s}, \quad (4.2)$$

$$R_u = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} R_u = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{TP_u}{TP_u + FN_u}. \quad (4.3)$$

where, in Eq. (4.2), we compute  $R_s$ , for the fixed seen class  $s \in \mathcal{S}$ , as the ratio between true positives  $TP_s$  and the total test examples of the class  $s$ , that is the sum of  $TP_s$  and the false negatives  $FN_s$  for that class. To obtain  $R_s$  from  $R_s$ ,  $s \in \mathcal{S}$ , we average  $R_s$  over the whole list of seen classes (having cardinality  $|\mathcal{S}|$ ). Analogous operations are carried out in Eq. (4.3) to compute  $R_u$ , but applied to unseen classes in  $\mathcal{U}$ , instead. The metrics  $H_{GZSL}$ ,  $R_s$  and  $R_u$  were proposed in [17] and adopted by state-of-the-art methods for their experimental validation [2; 59; 68; 72; 74; 77; 78; 87; 90; 91; 92].

In Generalized Zero-Shot Learning, given that both seen and unseen classes have to be reliably classified, it makes sense to have error metrics depending upon true positives and false negatives which are computed independently over seen and unseen classes and (harmonically) averaged in order to balance performance over these two sets of categories [17].

In Open Zero-Shot Learning, in order to break the closed-set assumption, we need to take into account also false positives  $FP$ . In fact,  $FP$  simulates cases where examples are predicted as if they belong to that class, albeit their actual ground-truth class is different. Please note that, since we cannot write explicit multi-class classification accuracy scores for the unknown classes  $\Omega$  - since we do not have anything describing them - we have to rely on false positives, for both seen and unseen classes ( $FP_s$ , for every  $s \in \mathcal{S}$ , and  $FP_u$ , for every  $u \in \mathcal{U}$ ), in



### 4.3 Problem Formulation

---

order to indirectly control the rejection performance. In other words, in order to quantitatively measure the performance of a predictor of seen and unseen classes  $\mathcal{S}$  and  $\mathcal{U}$ , which is also a rejector of unknown classes  $\Omega$ , we need to control  $FP_s$  and  $FP_u$ , for every  $s \in \mathcal{S}$  and  $u \in \mathcal{U}$ . This will reduce the possibility of wrongly associating generic unknown instances to any of the seen/unseen classes.

Obviously, the prior control on seen/unseen false positives has to be paired with penalization of “traditional” mis-classifications in a Generalized Zero-Shot Learning sense, since we do not want to gain in robustness towards unknown categories while forgetting how to predict seen or unseen classes. Therefore, we propose to measure performance in Open Zero-Shot Learning through the harmonic mean

$$\boxed{\mathcal{H}_{\text{OZSL}} = \frac{2F1_s F1_u}{F1_s + F1_u}} \quad (4.4)$$

of the  $F1$  scores  $F1_s$  and  $F1_u$ , over seen and unseen classes, defined as

$$F1_s = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} F1_s = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{2R_s P_s}{R_s + P_s}, \quad (4.5)$$

$$F1_u = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} F1_u = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{2R_u P_u}{R_u + P_u}. \quad (4.6)$$

In Eq. (4.5), for each seen class  $s \in \mathcal{S}$ , we compute the harmonic mean  $F1_s$  of  $R_s$ , defined as in Eq. (4.2), and the precision  $P_s$  relative to  $s$ . We have that  $P_s = \frac{TP_s}{TP_s + FP_s}$ , being defined as the ratio of the true positives  $TP_s$  for that class and the total test examples classified as belonging to that class, that is the sum of  $TP_s$  and false positives  $FP_s$ . We repeat the analogous operations over unseen classes to obtain  $F1_u$ , as in Eq. (4.6).

We claim that  $\mathcal{H}_{\text{OZSL}}$ , as defined in Eq. (4.4) extends the prior metric  $H_{\text{GZSL}}$  (in Eq. (4.1)) by preserving its property of evaluating a correct classification of seen and unseen categories. Concurrently, with  $\mathcal{H}_{\text{OZSL}}$ , we also inject false

positives, formalizing their addition using  $F1$  scores, for the sake of controlling any misclassification involving unknown classes: this is a computable proxy to evaluate performance on unknown classes.

## 4.4 Unknown Feature Generation

Feature generators for Generalized Zero-Shot Learning, such as [2] or [3], leverage the operative assumption of knowing the class embeddings even for the categories which are unseen at training time. Class embeddings are, in fact, adopted as conditioning factors inside GAN- [2; 4], VAE- [74; 90] or GAN+VAE-based methods [3; 78] to synthesize visual descriptors for the unseen classes. We cannot repeat the very same operation for unknown classes  $\Omega$  since we have no class embeddings, but we still need to generate visual features because we do not have them as well.

### 4.4.1 Direct Unknown Generation (DUG)

Exploiting the generative methods [2; 3; 4], is possible to train a generative method, trained only on seen categories, to be conditioned on semantic embeddings to generate corresponding visual features. Thus, once the training is complete, is possible to condition on the class embeddings of the unseen classes to generate unseen visual features. Once both the seen and unseen visual features are available, inspired by [118], we take advantage of the MixUp approach to directly generate visual features for the unknown categories.

That is, given two visual features  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , representative of different classes, we mix them with

$$\mathbf{x}_k = \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2, \quad (4.7)$$

where  $\lambda \in [0, 1]$  is sampled from a distribution  $\text{Beta}(\alpha, \beta)$ , and the unknown label

is assigned to  $\mathbf{x}_k$ .

The mixed features present mixed traits of the seen and unseen categories, belonging to any of them, and lie in regions of the feature space in between different classes. By labeling them as unknown, we can heuristically build borders for the classification regions for the seen and unseen classes and create a prior knowledge for classifying the unknown features.

### 4.4.2 Semantic Based Unknown Generation (SBUG)

Instead of directly using the visual feature space to generate the unknown features, a different approach that we investigate is to take advantage of the semantic embeddings to generate them. To this end, we propose to adopt a generative process to learn the distribution of the semantic space, as to learn the region of influence of seen and unseen class embeddings (blue and yellow balls in Fig. 4.3). So doing, we can map class embeddings into a transformed semantic space, and we claim that, inside it, we can generate class embeddings for the unknown classes by performing a mixing approach similar to the one presented in Section 4.4.1 Specifically, we sample the transformed semantic space “in between” the region of interest of seen and unseen classes, obtaining synthetic unknown class embeddings. Using them, we generate unknown visual features which help a classifier in rejecting unknown classes while still reliably classifying seen and unseen ones (from real seen and synthetic unseen visual features, respectively).

Thus, differently from DUG, where we can apply the unknown feature generation over an existing methodology, with SBUG we perform an end-to-end training together with the generative process to learn the mapping of the semantic embeddings in a new, more controllable, semantic space.

#### 4.4.2.1 VAcWGAN

We introduce a semantic sampler  $S$  which is responsible of learning first and second order statistics ( $\mu$  and  $\Sigma$ ) for each of the classes  $y$  whose semantic embedding is given (seen and unseen). Once trained, we sample a vector  $\mathbf{s}$  from a Gaussian distribution of mean  $\mu$  and covariance matrix  $\Sigma\Sigma^\top$ . The role of  $S$  is to transform the semantic space through a generative process, as the result of which, seen class embeddings  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k$ , and unseen ones  $\mathcal{C}_{k+1}, \mathcal{C}_{k+2}, \dots, \mathcal{C}_{k+u}$  are mapped into regions of influence. That is, they are mapped into  $\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_k$  (light blue balls in Fig. 4.3) and  $\mathcal{N}_{k+1}, \mathcal{N}_{k+2}, \dots, \mathcal{N}_{k+u}$  (yellow balls in Fig. 4.3). We model  $\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_k, \mathcal{N}_{k+1}, \mathcal{N}_{k+2}, \dots, \mathcal{N}_{k+u}$  as Gaussian distributions and we use them to sample the conditioning factor  $\mathbf{s}$  which, paired to a random noise vector  $\mathbf{z}$  is passed to a Wasserstein GAN. This GAN is trained to generate synthetic visual features  $\tilde{\mathbf{x}}$  by making them indistinguishable from the real seen features  $\mathbf{x}$  extracted by an ImageNet pre-trained ResNet-101 model.

We call the aforementioned architecture variationally-conditioned Wasserstein GAN (VAcWGAN), which is built over the following optimization:  $\min_{G,S} \max_D \mathcal{L}$ , where

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{s}) &= L^{\text{real}}(\mathbf{x}, \mathbf{s}) - L^{\text{fake}}(\tilde{\mathbf{x}}, \mathbf{s}) \\ &= \mathbb{E}_{\mathbf{x} \sim \text{real}} [D(\mathbf{x}, \mathbf{s})] - \mathbb{E}_{\tilde{\mathbf{x}} \sim \text{gen}} [D(\tilde{\mathbf{x}}, \mathbf{s})]. \end{aligned} \quad (4.8)$$

In Eq. (4.8),  $\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{s})$  attempts to align the Wasserstein (Earth Mover) distance [44] between the distributions of synthesized features  $\tilde{\mathbf{x}}$  over the distribution of the real ones  $\mathbf{x}$ . We introduce two auxiliary losses for VAcWGAN by jointly considering a standard gradient penalty term [45]

$$\mathcal{R}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{s}) = \mathbb{E}_{t \in [0,1]} [(\|\nabla D(t\mathbf{x} + (1-t)\tilde{\mathbf{x}}, \mathbf{s})\|_2 - 1)^2]$$

## 4.4 Unknown Feature Generation

---

which is commonly acknowledged to regularize the whole generation process, increasing computational stability [45]. We used a cross-entropy classification loss [2]

$$C(\tilde{\mathbf{x}}) = -\mathbb{E}_{\tilde{\mathbf{x}} \sim \text{gen}} [\log p(y|\tilde{\mathbf{x}})] \quad (4.9)$$

which constrains the softmax probability  $p$  of classifying  $\tilde{\mathbf{x}}$  to belong to the class  $y$ : it has to match the prediction done on  $\tilde{\mathbf{x}}$  when generated from the class embedding  $\mathcal{C}_y$  relative to the class  $y$ .

The pseudocode to train VAcWGAN is provided in Alg. 1.

---

### Algorithm 1: Training VAcWGAN

---

```

1 Randomly initialize  $S, G$  and  $D$  ;
2 Generate  $\tilde{\mathbf{x}}$  and pre-train the softmax classifier  $p$  while not converged do
3   for  $i \leftarrow 1$  to  $M$  do
4     | update  $D$  using  $\mathcal{L}$  and  $\mathcal{R}$ 
5   end
6   Synthesized unseen features  $\tilde{\mathbf{x}}$  ;
7   Update  $G$  using  $L^{\text{fake}}, \mathcal{R}$  and  $C$ ;
8   Update  $S$  using  $L^{\text{fake}}, \mathcal{R}$  and  $C$ ;
9 end

```

---

#### 4.4.2.2 Implementation details

We implement  $G$ ,  $D$  and  $S$  as single hidden layer neural networks with hidden layer of size 4096 for  $G$  and  $D$  and 2048 for  $S$  with leaky ReLU activation for all.  $S$  takes as input the class embeddings  $\mathcal{C}$  and gives as output mean and gives as output mean vector  $\mu$  and  $\log(\sqrt{\sigma})$  of the same size of  $\mathcal{C}$ .  $G$  takes as input the vector  $\mathbf{s}$ , sampled from the Gaussian distribution defined by  $\mu$  and  $\log(\sqrt{\sigma})$  concatenated with a noise vector  $\mathbf{z}$  of the same size of  $\mathbf{s}$  sampled from a multivariate normal distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ , where  $\mathbf{0}$  a vector of zeros and  $\mathbf{I}$  and identity matrix, and output a visual feature vectors (of size 2048 and ReLU activation).  $D$  takes

as input visual feature vectors with the related class embedding  $\mathcal{C}$  and output an unconstrained real number. To compute the regularization classification loss we directly classify the synthesized visual features with a pre-trained softmax.  $M$  of Alg. 1 (in the Chapter) is fixed to 5. Adam [119] is used as optimizer.

#### 4.4.2.3 Unknown Generation.

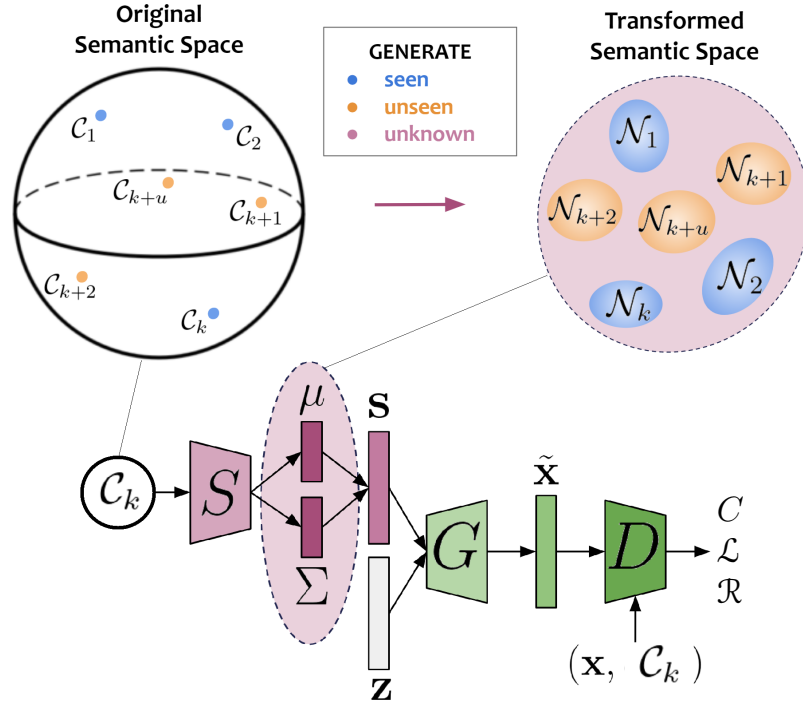


Figure 4.3: Using VAcWGAN, we generate unknown class embeddings (in a transformed semantic space) from which, in turn synthetic unknown visual features can be generated.

We train VAcWGAN using *seen data only*. In addition to generating unseen visual features (as commonly done in Generalized Zero-Shot Learning, see Section 4.2), we can also generate the unknown with a two-stages process. Given the generative process that VAcWGAN endow on class embeddings, we estimate the region of interest  $\mathcal{N}_1 \cup \mathcal{N}_2 \cup \dots \cup \mathcal{N}_k \cup \mathcal{N}_{k+1} \cup \mathcal{N}_{k+2} \cup \dots \cup \mathcal{N}_{k+u}$  of both seen and unseen

classes (in a transformed semantic space). We can exploit the complementarity of it (*i.e.*, the pink region in Figure 4.3) to sample class embeddings that lie in the new semantic space in the regions in between the seen and unseen classes by mixing samples of seen and unseen class embeddings.

Specifically, we sample two semantic embeddings for two different classes  $\mathcal{C}_i$  and  $\mathcal{C}_j$ , sample accordingly to the regions of interest  $\mathbf{s}_i \sim \mathcal{N}_i$  and  $\mathbf{s}_j \sim \mathcal{N}_j$ , and then we mix them with

$$\mathbf{s}_k = \lambda \mathbf{s}_i + (1 - \lambda) \mathbf{s}_j, \quad (4.10)$$

where  $\lambda \in [0, 1]$  is sampled from a distribution  $\text{Beta}(\alpha, \beta)$ , and the unknown label is assigned to  $\mathbf{s}_k$ . Once unknown class embeddings are sampled, they can be used as a conditioning factor to generate visual features that can be ascribed to the unknown classes.

## 4.5 Experiments

In order to understand how difficult Open Zero-Shot Learning is, in this Section, we inspect the performance achievable by combining state-of-the-art feature generators ([2; 3] for ZSL, combining them with a state-of-the-art classifier for open recognition: Openmax [1]).

In Figure 4.4, we compare a standard softmax (in blue) vs. Openmax (in red), using a CLSWGAN [2] for unseen (but not unknown) feature generation. On average, we do not register a sharp overall advantage of openmax versus softmax (only +0.6% boost in precision and +0.9% for recall given by Openmax tuned with tail size 2). In principle, openmax is theoretically superior to a softmax operator, since it is capable of performing rejection. However, experimentally, we did not register a direct consequence in a superior classification performance. In fact, the recall of Openmax in rejecting the unknown ( $R_\Omega = 22.12\%$ ) is not

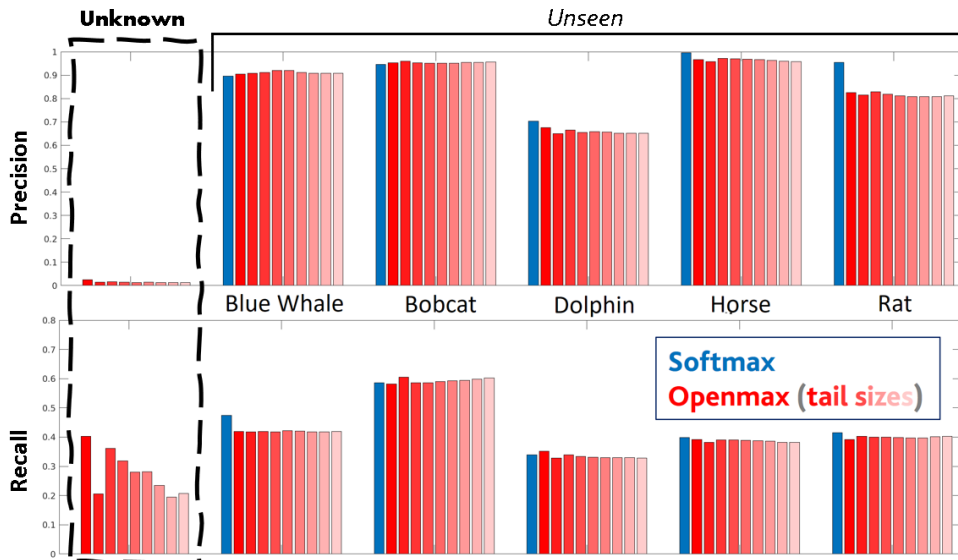


Figure 4.4: Precision and Recall of CLSWGAN [2] combined with either Softmax or Openmax [1] on the AWA dataset. Openmax has been tuned using different tail sizes (ranging from 2 - darkest red - to 10 - lighter red). When adopting state-of-the-art solutions (like [1]) to cope with the unknown, we argue that the joint presence of unseen classes (which we do not have to forget) prevents Openmax to reliably rejecting the unknown - as it appears to be able to if we remove unseen classes (see [1]). We perceive this as evidence of the challenges related to Open Zero-Shot Learning.

so dissimilar to the recall values scored on some unseen classes (*e.g.*, Horse or Bobcat in Fig. 4.4). As shown in the literature [1], Openmax is arguably a state-of-the-art method to perform rejection, while also recognizing seen classes only. However, a plain transfer of Openmax from its original framework to Open Zero-Shot Learning (in which, unseen classes have to be recognized *as well*) is not enough to solve the problem. We deem this as evidence for the intrinsic difficulty of Open Zero-Shot Learning which appears as arguably hard - and thus intriguing.

Even resorting to a better feature generator is not enough to solve the problem, as we show in Table 4.1. Therein, we provide a comparison between the  $F1_{\Omega}$  score computed over unknown classes, pretending to treat all unknown classes into a



## 4.5 Experiments

tf-VAEGAN [3]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	bobcat	giraffe	horse	sheep	$F1_u$	
AWA	Softmax	0.00%	0.00%	0.00%	84.90%	87.61%	39.40%	59.80%	72.22%
	Openmax	18.81%	58.24%	28.43%	92.66%	87.56%	37.38%	48.94%	76.04%
CLSWGAN [2]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	blue whale	bobcat	dolphin	rat	$F1_u$	
AWA	Softmax	0.00%	0.00%	0.00%	72.07%	72.36%	45.73%	56.97%	70.42%
	Openmax	22.45%	70.87%	34.10%	77.35%	75.26%	46.25%	55.75%	74.90%
tf-VAEGAN [3]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	green violetear	scarlet tanager	tree sparrow	yellowthroat	$F1_u$	
CUB	Softmax	0.00%	0.00%	0.00%	89.55%	88.06%	26.67%	51.67%	67.15%
	Openmax	0.80%	40.00%	1.57%	100.00%	98.33%	12.31%	18.90%	69.43%
CLSWGAN [2]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	bl. cormorant	red c woodp	orange warb	mockingbird	$F1_u$	
CUB	Softmax	0.00%	0.00%	0.00%	83.05%	96.55%	31.67%	13.51%	64.45%
	Openmax	3.35%	70.16%	6.40%	69.49%	89.66%	26.39%	17.22%	68.53%
tf-VAEGAN [3]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	purple cone	tigerlily	pink prim	sweetpea	$F1_u$	
FLO	Softmax	0.00%	0.00%	0.00%	80.65%	93.33%	45.00%	42.86%	65.14%
	Openmax	10.16%	65.17%	17.58%	88.24%	73.68%	37.50%	20.51%	69.78%
CLSWGAN [2]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	purple cone	camellia	buttercup	azalea	$F1_u$	
FLO	Softmax	0.00%	0.00%	0.00%	88.24%	82.14%	30.90%	46.43%	52.56%
	Openmax	18.32%	81.36%	29.91%	80.65%	80.36%	32.92%	44.64%	53.80%
tf-VAEGAN [3]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	hoodoo	fishpond	bow wind. ind.	elevator	$F1_u$	
SUN	Softmax	0.00%	0.00%	0.00%	85.00%	85.00%	50.00%	35.00%	56.33%
	Openmax	2.06%	43.35%	3.92%	95.00%	85.00%	29.41%	20.00%	61.68%
CLSWGAN [2]	$P_\Omega$	$R_\Omega$	$F1_\Omega$	car seat	church indoor	field cult.	ballroom	$F1_u$	
SUN	Softmax	0.00%	0.00%	0.00%	86.36%	70.59%	22.67%	22.92%	50.44%
	Openmax	6.94%	58.99%	12.43%	94.44%	75.00%	24.14%	38.89%	53.76%

Table 4.1: Baseline methods for Open Zero-Shot Learning evaluated on their capability of rejecting unknown (treated as a separated class for which precision  $P_\Omega$ , recall  $R_\Omega$ , and F1  $F1_\Omega$  scores can be computed. We also focus here on classifying unseen classes, reporting the average F1 score  $F1_u$  over them, while also reporting the per-class F1 score for two exemplar classes whose performance is above the mean (6th, 7th columns, marked in green), and two classes that are below it (8th, 9th columns, marked in red). We observe that, generically, Openmax achieves high recall and low precision. The softmax is not capable of rejecting, therefore  $P_\Omega = R_\Omega = F1_\Omega = 0\%$ .

macro-container called “unknown” (while in principle unknown instances belong to potentially infinite different unknown categories). In addition, we also check  $F1_u$ , the F1 score over unseen classes only. While exploiting a better model than tf-VAEGAN, we can surely always state that Openmax yields a better  $F1_u$  with respect to CLSWGAN with Openmax (76.04% vs. 74.90% on AWA, 69.43% vs. 68.53% on CUB, 69.78% vs. 53.80% on FLO and 61.68% vs. 53.76% on SUN), while also improving tf-VAEGAN with softmax (improving  $F1_u$  by +4%

on AWA, +2% on CUB, +5% on FLO and +6% on SUN). But, this result comes at the price of loosing in  $F1_{\Omega}$ , whose performance is much higher when using CLSWGAN as opposed to tf-VAEGAN (-6% on AWA, -4% on CUB, -12% on FLO and -8% on SUN).

Apparently, taking existing feature generators for ZSL (CLSWGAN [2] or tf-VAEGAN [3]) and combining them with state-of-the-art methods in open recognition (like openmax [1]) is not enough to solve Open Zero-Shot Learning which appears as an intriguing problem. To start solving it, in the next Section, we evaluate the effect of performing unknown feature generation.

#### 4.5.1 Open Zero-Shot Learning through Unknown Generation

In Table 4.2 we perform an experimental evaluation between the two strategies of unknown feature generation we presented: DUG (Direct Unknown Generation, Sec. 4.4.1) and SBUG (Semantic Based Unknown Generation, Sec. 4.4.2). We combined DUG and SBUG separately and/or jointly to the VAcWGAN architecture (Sec. 4.4). In the sharp majority of the cases, doing unknown feature generation (with either DUG, SBUG or DUG+SBUG) is better than not doing it. We deem this result highly non-trivial: by attempting to learn how to synthesize unknown descriptors, we are simultaneously better shaping the region of interest of seen and unseen classes, so that the  $F1_u$  and  $F1_s$  metrics often improve (and so happens for their harmonic mean  $\mathcal{H}_{\text{OZSL}}$  as well). For instance, the unknown feature generation improves by +1.48%, +1.02% and +0.18% the performance of WAcWGAN on AWA, CUB and SUN respectively, while considering the  $\mathcal{H}_{\text{OZSL}}$  metric and the SBUG, DUG + SBUG and DUG techniques, respectively.

Over DUG and SBUG, the former is advantageous over the latter because it acts directly on the visual space (so that the feature generator has not to be

## 4.5 Experiments

	AWA			CUB			FLO			SUN		
	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$
VAcWGAN	50.31%	64.84%	56.66%	45.08%	49.23%	47.34%	<b>47.68%</b>	72.69%	<b>57.59%</b>	<b>38.05%</b>	37.33%	37.68%
VAcWGAN + DUG	<b>51.83%</b>	65.18%	57.74%	45.48%	49.98%	47.63%	46.25%	<b>73.30%</b>	56.71%	33.87%	<b>38.37%</b>	<b>37.68%</b>
VAcWGAN + SBUG	50.62%	<b>68.28%</b>	<b>58.14%</b>	45.59%	<b>51.42%</b>	48.04%	46.01%	69.40%	55.33%	37.65%	38.07%	35.98%
VAcWGAN + DUG + SBUG	51.63%	66.20%	58.01%	<b>45.76%</b>	51.28%	<b>48.36%</b>	47.24%	72.00%	57.05%	34.83%	38.53%	36.59%

Table 4.2: Direct and semantic unknown generation (DUG and SBUG) for the WAcWGAN model (check Sec. 4.4.2).

	AWA			CUB			FLO			SUN		
	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$	$F1_u$	$F1_s$	$\mathcal{H}_{OZSL}$
CSLWGAN	52.37%	65.52%	58.21%	47.01%	52.77%	49.73%	48.47%	76.08%	59.22%	36.31%	38.77%	37.50%
CLSWGAN + DUG	<b>57.34%</b>	<b>67.41%</b>	<b>61.97%</b>	<b>47.76%</b>	<b>53.29%</b>	<b>50.37%</b>	<b>49.34%</b>	<b>76.48%</b>	<b>59.98%</b>	<b>36.92%</b>	<b>39.64%</b>	<b>38.23%</b>
tf-VAEGAN	55.49%	71.47%	62.48%	<b>52.24%</b>	56.62%	54.34%	<b>54.78%</b>	80.00%	65.03%	43.00%	<b>41.09%</b>	42.02%
tf-VAEGAN + DUG	<b>60.56%</b>	<b>71.73%</b>	<b>65.68%</b>	51.60%	<b>57.92%</b>	<b>54.58%</b>	54.35%	<b>81.33%</b>	<b>65.15%</b>	<b>46.53%</b>	40.76%	<b>42.10%</b>
CEZSL	51.70%	<b>71.66%</b>	60.07%	39.43%	56.83%	46.56%	39.21%	<b>85.36%</b>	53.74%	33.35%	<b>30.83%</b>	32.04%
CEZSL + DUG	<b>55.76%</b>	71.30%	<b>62.58%</b>	<b>40.52%</b>	<b>57.54%</b>	<b>47.55%</b>	<b>40.26%</b>	85.25%	<b>54.69%</b>	<b>35.17%</b>	30.15%	<b>32.47%</b>

Table 4.3: The effect of direct unknown generation (DUG) applied on three benchmark methods: CSLWGAN [2], tf-VAEGAN [3] and CEZSL [4] on the proposed OZSL splits for AWA [5], CUB [6], FLO [7] and SUN [8] datasets.

re-trained for unknown synthesis, but can be adapted to it). In view of this consideration, we can apply unknown feature generation to three state-of-the-art approaches for ZSL, better tailoring them towards the open ZSL regime. Namely, we consider the following methods: the Wasserstein generative adversarial network conditioned on class embeddings (CLSWGAN) [2] and its extension tf-VAEGAN [3] in which this architecture is paired with a variational auto-encoder to boost the generation stage. We also considered the usage of contrastive learning as adopted in CEZSL [4] in tandem with adversarial training. We combine CLSWGAN, tf-VAEGAN and CEZSL with the direct unknown generation that we presented in Sec. 4.4.1 and dubbed here ‘‘DUG’’ for brevity. As we show in Table 4.3, the adoption of DUG is always able to improve in performance all the considered baseline approaches with respect to the  $\mathcal{H}_{OZSL}$  metric (*e.g.*, +3.76% on AWA for CLSWGAN, +0.73% on SUN for CLSWGAN, +0.99% on CUB for CEZSL and +3.2% on AWA for tf-VAEGAN). Again, we interpret the consistent improvements that we registered as evidence for the effectiveness of performing unknown feature generation for Open Zero-Shot Learning.

## 4.6 Conclusions and Future Work

In this Chapter, we proposed a novel paradigm, called Open Zero-Shot Learning, extending traditional ZSL frameworks towards the additional rejection of unknown categories (neither visually nor semantically described) while still recognizing seen and unseen classes. Using the experimental protocols and error metrics that we proposed, our experimental findings suggest that attempting to synthesize unknown descriptors (to be rejected) seems a viable solution for Open Zero-Shot Learning.

Future works will be aimed at adopting techniques from out-of-domain generalization to better achieve the way we explore the semantic/visual spaces while seeking better strategies to generate the unknown.

# Chapter 5

## Conclusion

In this thesis, we introduced the Zero-Shot Learning classification problem and its different setups. Among the real-world scenarios, where annotating big corpora of data that are balanced in the classes to be predicted is commonly not feasible, Zero-Shot Learning analyze the extreme case of them. Zero-Shot Learning considers two disjoint classes: the seen classes, for which label data are available, and the unseen classes, for which visual data are not available at all (inductive Zero-Shot Learning) or available but without annotation (transductive Zero-Shot Learning). Class embeddings are used to perform knowledge transfer from the seen to the unseen domain. Class embeddings are class level semantic information that needs to be shared and discriminative across the classes.

Generative Zero-Shot Learning is a competitive and challenging scenario in which the model has to balance the performance on both, the seen and the unseen classes. To address the data unbalance and to avoid the classifier being biased towards predicting the seen classes, recently, generative approaches have been proposed.

The generative approaches take advantage of deep generative neural networks that, conditioned on the class embeddings, produce synthetic labeled data for the

---

unseen classes.

We proposed a generative solution focusing on the transductive Generalized Zero-Shot Learning setup, that is we can take advantage of unlabeled data. We propose the decoupled-features generation to alleviate the seen to unseen domain-shift problem and improve the data generation. With our decoupled feature generation we improve the control in the two tasks of capturing the domain distribution and translating the semantic content of the class embeddings into visual features. We implemented the decoupled feature generation with our DecGAN architecture and achieved with it state-of-the-art performance on different transductive Generalized Zero-Shot Learning benchmark datasets.

We additionally proposed a further extension of Generalized Zero-Shot Learning setup that is more competitive and more oriented towards real-world scenarios and named it Open Zero-Shot Learning. In fact, obtaining the class embeddings can be as well difficult and costly as for the annotated visual features. Thus, we removed the constraint of having all the possible unseen classes represented by the class embeddings. Consequently, we have three set of classes: the seen, the unseen and the unknown. The seen and the unseen classes are defined as in Zero-Shot Learning, while the unknown classes are the ones for which we have neither the visual features nor the class embeddings. The task in Open Zero-Shot Learning is to correctly classify the seen and the unseen classes while rejecting the unknown instances. We formalized the problem, proposed publicly available benchmark datasets and evaluate different baselines. Additionally, we tackled the problem with novel unseen and unknown synthetic data generation with two different strategies. In particular we propose and implement with VAcWGAN our idea of generating unknown class embeddings.

# References

- [1] A. Bendale and T. E. Boult, “Towards open set deep networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1563–1572, 2016. vi, vii, 19, 47, 48, 50, 60, 61, 63
- [2] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, “Feature generating networks for zero-shot learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. vii, ix, 22, 25, 33, 35, 42, 43, 46, 47, 48, 49, 52, 53, 55, 58, 60, 61, 62, 63, 64
- [3] S. Narayan, A. Gupta, F. S. Khan, C. G. Snoek, and L. Shao, “Latent embedding feedback and discriminative features for zero-shot classification,” in *The European Conference on Computer Vision (ECCV)*, 2020. ix, 5, 22, 23, 24, 43, 48, 55, 60, 62, 63, 64
- [4] Z. Han, Z. Fu, S. Chen, and J. Yang, “Contrastive embedding for generalized zero-shot learning,” in *CVPR*, 2021. ix, 22, 46, 48, 49, 55, 64
- [5] C. Lampert, H. Nickisch, and S. Harmeling, “Learning to detect unseen object classes by between-class attribute transfer,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. ix, 52, 64, 89
- [6] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and

## REFERENCES

---

- P. Perona, “Caltech-UCSD Birds 200,” Tech. Rep. CNS-TR-2010-001, California Institute of Technology, 2010. ix, 10, 27, 36, 37, 52, 64, 89, 91
- [7] M.-E. Nilsback and A. Zisserman, “A visual vocabulary for flower classification,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. ix, 36, 37, 52, 64, 88, 89, 97
- [8] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. ix, 27, 36, 37, 52, 64, 89, 94
- [9] R. Salakhutdinov, A. Torralba, and J. B. Tenenbaum, “Learning to share visual appearance for multiclass object detection,” *CVPR 2011*, pp. 1481–1488, 2011. 2, 24
- [10] Y. Ganin and V. S. Lempitsky, “Unsupervised domain adaptation by back-propagation,” *ArXiv*, vol. abs/1409.7495, 2015. 2
- [11] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *ArXiv*, vol. abs/1412.3474, 2014. 2
- [12] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, “Undoing the damage of dataset bias,” in *Computer Vision – ECCV 2012*. 2
- [13] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006. 3
- [14] M. Fink, “Object classification from a single example utilizing class relevance metrics,” in *NIPS*, 2004. 3



## REFERENCES

---

- [15] H. Larochelle, D. Erhan, and Y. Bengio, “Zero-data learning of new tasks.,” in *Conference on Artificial Intelligence (AAAI)*, AAAI, 2008. 3, 24, 45
- [16] 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 (CVPR)*, IEEE, 2009. 3, 24, 45, 51
- [17] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, “Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly,” *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 3, 13, 22, 25, 28, 37, 42, 43, 48, 49, 51, 52, 53, 89
- [18] Z. Wan, D. Chen, Y. Li, X. Yan, J. Zhang, Y. Yu, and J. Liao, “Transductive zero-shot learning with visual structure constraint,” in *Advances in Neural Information Processing Systems 32*, pp. 9972–9982, 2019. 5, 22, 23, 24, 28, 41, 43
- [19] M. Ye and Y. Guo, “Progressive ensemble networks for zero-shot recognition,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11720–11728, 2019. 5, 22, 23, 24, 28, 41, 43, 44
- [20] E. Kodirov, T. Xiang, and S. Gong, “Semantic autoencoder for zero-shot learning,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4447–4456, 2017. 5, 22, 24
- [21] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *EMNLP*, 2014. 10
- [22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,”

## REFERENCES

---

- in *Advances in neural information processing systems*, pp. 3111–3119, 2013. 10
- [23] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, “Fasttext.zip: Compressing text classification models,” *ArXiv*, vol. abs/1612.03651, 2016. 10
- [24] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *ICLR*, 2013. 10
- [25] 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 (CVPR)*, June 2016. 11, 37, 88
- [26] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997. 11
- [27] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, “Learning to learn by gradient descent by gradient descent,” in *NIPS*, 2016. 11
- [28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 12
- [29] 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*, vol. 1, no. 4, pp. 541–551, 1989. 12
- [30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information*

## REFERENCES

---

- Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012. 12
- [31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015. 13
- [32] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2015. 13
- [33] 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*, pp. 770–778, 2016. 13, 37
- [34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016. 13
- [35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. 13
- [36] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, 2014. 15, 16, 25, 29
- [37] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, “Improved techniques for training gans,” in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), 2016. 15

## REFERENCES

---

- [38] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *CoRR*, vol. abs/1511.06434, 2016. 15
- [39] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *ArXiv*, vol. abs/1809.11096, 2019. 15
- [40] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, 2019. 15
- [41] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017. 15
- [42] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,” *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5908–5916, 2017. 15
- [43] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” 2017. 16
- [44] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” *ArXiv*, vol. abs/1701.07875, 2017. 16, 25, 29, 32, 57
- [45] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *NIPS*, 2017. 16, 25, 29, 32, 33, 35, 48, 57, 58
- [46] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *CoRR*, vol. abs/1312.6114, 2013. 17, 29

## REFERENCES

---

- [47] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *ArXiv*, vol. abs/1411.1784, 2014. 17
- [48] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi, “Generative openmax for multi-class open set classification,” 07 2017. 19, 50
- [49] M. Hassen and P. K. Chan, “Learning a neural-network-based representation for open set recognition,” in *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 154–162, SIAM, 2020. 19
- [50] A. Rozsa, M. Günther, and T. Boult, “Adversarial robustness: Softmax versus openmax,” 01 2017. 19
- [51] C. H. Lampert, H. Nickisch, and S. Harmeling, “Attribute-based classification for zero-shot visual object categorization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 453–465, 2014. 20, 45
- [52] D. Jayaraman and K. Grauman, “Zero-shot recognition with unreliable attributes,” in *NIPS*, 2014. 20
- [53] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, “Label-embedding for image classification,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 7, pp. 1425–1438, 2015. 21, 28, 41, 42, 43
- [54] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov, “Devise: A deep visual-semantic embedding model,” in *Advances in Neural Information Processing Systems* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013. 21

- 
- [55] B. Romera-Paredes and P. Torr, “An embarrassingly simple approach to zero-shot learning,” in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2152–2161, 2015. 21, 42, 43
- [56] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele, “Evaluation of output embeddings for fine-grained image classification,” 06 2015. 21
- [57] Y. Li, Z. Jia, J. Zhang, K. Huang, and T. Tan, “Deep semantic structural constraints for zero-shot learning,” in *AAAI*, 2018. 21
- [58] Y. Xian, Z. Akata, G. Sharma, Q. N. Nguyen, M. Hein, and B. Schiele, “Latent embeddings for zero-shot classification,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 69–77, 2016. 21, 42
- [59] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, and Z. Akata, “Generalized zero- and few-shot learning via aligned variational autoencoders,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 21, 25, 42, 43, 46, 48, 49, 52, 53
- [60] 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 *ICLR*, 2014. 21
- [61] Z. Zhang and V. Saligrama, “Zero-shot learning via semantic similarity embedding,” *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4166–4174, 2015. 21
- [62] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha, “Synthesized classifiers for zero-shot learning,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5327–5336, 2016. 21, 42, 43

## REFERENCES

---

- [63] Y. Annadani and S. Biswas, “Preserving semantic relations for zero-shot learning,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7603–7612, 2018. 21
- [64] H. Jiang, R. Wang, S. Shan, and X. Chen, “Learning class prototypes via structure alignment for zero-shot recognition,” *ArXiv*, vol. abs/1807.09123, 2018. 21
- [65] J. Song, C. Shen, J. Lei, A. Zeng, K. Ou, D. Tao, and M. Song, “Selective zero-shot classification with augmented attributes,” in *ECCV*, 2018. 21
- [66] P. Peng, Y. Tian, T. Xiang, Y. Wang, and T. Huang, “Joint learning of semantic and latent attributes,” in *Computer Vision – ECCV 2016*. 21
- [67] V. K. Verma and P. Rai, “A simple exponential family framework for zero-shot learning,” in *ECML/PKDD*, 2017. 21, 23, 28, 41, 42, 43
- [68] R. Felix, V. B. Kumar, I. Reid, and G. Carneiro, “Multi-modal cycle-consistent generalized zero-shot learning,” in *The European Conference on Computer Vision (ECCV)*, 2018. 22, 25, 34, 42, 43, 46, 48, 49, 52, 53
- [69] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017. 22
- [70] A. Pambala, T. Dutta, and S. Biswas, “Generative model with semantic embedding and integrated classifier for generalized zero-shot learning,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020. 22

- 
- [71] Z. Chen, J. Li, Y. Luo, Z. Huang, and Y. Yang, “Canzsl: Cycle-consistent adversarial networks for zero-shot learning from natural language,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020. 22
- [72] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, and Z. Huang, “Leveraging the invariant side of generative zero-shot learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 22, 25, 46, 48, 49, 52, 53
- [73] Z. Han, Z. Fu, and J. Yang, “Learning the redundancy-free features for generalized zero-shot object recognition,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 22
- [74] A. Mishra, M. S. K. Reddy, A. Mittal, and H. A. Murthy, “A generative model for zero shot learning using conditional variational autoencoders,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2269–22698, 2018. 22, 25, 46, 48, 49, 53, 55
- [75] R. Keshari, R. Singh, and M. Vatsa, “Generalized zero-shot learning via over-complete distribution,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 22
- [76] Z. Chen, Y. Luo, R. Qiu, S. Wang, Z. Huang, J. Li, and Z. Zhang, “Semantics disentangling for generalized zero-shot learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8712–8720, October 2021. 22
- [77] R. Gao, X. Hou, J. Qin, J. Chen, L. Liu, F. Zhu, Z. Zhang, and L. Shao, “Zero-vae-gan: Generating unseen features for generalized and transduc-



- 
- tive zero-shot learning,” *IEEE Transactions on Image Processing*, vol. 29, pp. 3665–3680, 2020. 22, 23, 25, 28, 29, 41, 43, 44, 46, 48, 49, 52, 53
- [78] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, “F-VAEGAN-D2: A Feature Generating Framework for Any-Shot Learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 22, 23, 25, 28, 29, 37, 41, 42, 43, 46, 48, 49, 52, 53, 55
- [79] S. Chen, W. Wang, B. Xia, Q. Peng, X. You, F. Zheng, and L. Shao, “Free: Feature refinement for generalized zero-shot learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 122–131, October 2021. 22
- [80] M. Ye and Y. Guo, “Zero-shot classification with discriminative semantic representation learning,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5103–5111, 2017. 22, 28, 41, 42, 43
- [81] Y. Fujiwara and G. Irie, “Efficient label propagation,” in *ICML*, 2014. 22, 28
- [82] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, “Transductive multi-view zero-shot learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 2332–2345, 2015. 22
- [83] “Rethinking zero-shot learning: A conditional visual classification perspective,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 23
- [84] L. Zhang, P. Wang, L. Liu, C. Shen, W. Wei, Y. Zhang, and A. van den Hengel, “Towards effective deep embedding for zero-shot learning,” *ArXiv*, vol. abs/1808.10075, 2018. 23, 28, 41, 43

- 
- [85] A. Zhao, M. Ding, J. Guan, Z. Lu, T. Xiang, and J.-R. Wen, “Domain-invariant projection learning for zero-shot recognition,” in *NeurIPS*, 2018. 23, 28, 41, 43
- [86] J. Song, C. Shen, Y. Yang, Y. P. Liu, and M. Song, “Transductive unbiased embedding for zero-shot learning,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1024–1033, 2018. 23
- [87] M. B. Sariyildiz and R. G. Cinbis, “Gradient matching generative networks for zero-shot learning,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2163–2173, 2019. 23, 25, 28, 29, 41, 43, 46, 48, 49, 52, 53
- [88] J. Wu, T. Zhang, Z.-J. Zha, J. Luo, Y. Zhang, and F. Wu, “Self-supervised domain-aware generative network for generalized zero-shot learning,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 23
- [89] A. Paul, N. C. Krishnan, and P. Munjal, “Semantically aligned bias reducing zero shot learning,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7049–7058, 2019. 23
- [90] G. Arora, V. K. Verma, A. Mishra, and P. Rai, “Generalized zero-shot learning via synthesized examples,” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 25, 42, 43, 46, 48, 49, 52, 53, 55
- [91] H. Huang, C. Wang, P. S. Yu, and C.-D. Wang, “Generative dual adversarial network for generalized zero-shot learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 25, 46, 48, 49, 52, 53

- 
- [92] Y. Zhu, J. Xie, B. Liu, and A. Elgammal, “Learning feature-to-feature translator by alternating back-propagation for generative zero-shot learning,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 25, 46, 48, 49, 52, 53
- [93] I. J. Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” *ArXiv*, vol. abs/1701.00160, 2017. 28, 29
- [94] N. Bodla, G. Hua, and R. Chellappa, “Semi-supervised fusedgan for conditional image generation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 669–683, 2018. 34
- [95] W. Scheirer, A. Rocha, A. Sapkota, and T. Boult, “Towards open set recognition.,” *IEEE transactions on pattern analysis and machine intelligence*, 11 2012. 47, 50
- [96] 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*, pp. 1004–1013, 2018. 49
- [97] Y. Shen, J. Qin, L. Huang, L. Liu, F. Zhu, and L. Shao, “Invertible zero-shot recognition flows,” in *European Conference on Computer Vision*, pp. 614–631, Springer, 2020. 49
- [98] F. Marmoreo, J. Cavazza, and V. Murino, “Transductive zero-shot learning by decoupled feature generation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3109–3118, 2021. 49
- [99] M. Mancini, M. F. Naeem, Y. Xian, and Z. Akata, “Open world compositional zero-shot learning,” in *CVPR*, 2021. 49

## REFERENCES

---

- [100] W. J. Scheirer, L. P. Jain, and T. E. Boult, “Probability models for open set recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014. 50
- [101] A. Bendale and T. Boult, “Towards open world recognition,” 12 2014. 50
- [102] L. P. Jain, W. Scheirer, and T. Boult, “Multi-class open set recognition using probability of inclusion,” in *ECCV*, 2014. 50
- [103] H. Cevikalp, B. Triggs, and V. Franc, “Face and landmark detection by using cascade of classifiers,” *10th IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 1–7, 01 2013. 50
- [104] H. Cevikalp, “Best fitting hyperplanes for classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1–1, 07 2016. 50
- [105] M. Scherreik and B. Rigling, “Open set recognition for automatic target classification with rejection,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, pp. 632–642, 04 2016. 50
- [106] H. Cevikalp and B. Triggs, “Polyhedral conic classifiers for visual object detection and classification,” pp. 4114–4122, 07 2017. 50
- [107] H. Cevikalp and H. Yavuz, “Fast and accurate face recognition with image sets,” pp. 1564–1572, 10 2017. 50
- [108] H. Zhang and V. Patel, “Sparse representation-based open set recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1–1, 09 2016. 50

## REFERENCES

---

- [109] P. Júnior, R. Souza, R. Werneck, B. Stein, D. Pazinato, W. Almeida, O. Pennatti, R. Torres, and A. Rocha, “Nearest neighbors distance ratio open-set classifier,” *Machine Learning*, vol. 106, pp. 1–28, 03 2017. 50
- [110] E. Rudd, L. Jain, W. Scheirer, and T. Boult, “The extreme value machine,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, 06 2015. 50
- [111] E. Vignotto and S. Engelke, “Extreme value theory for open set classification - gpd and gev classifiers,” 08 2018. 50
- [112] G. Fei and B. Liu, “Breaking the closed world assumption in text classification,” pp. 506–514, 01 2016. 50
- [113] R. Vareto, S. Silva, F. Costa, and W. Schwartz, “Towards open-set face recognition using hashing functions,” 10 2017. 50
- [114] M. Córdova, P. Júnior, A. Rocha, and R. Torres, “Data-fusion techniques for open-set recognition problems,” *IEEE Access*, vol. 6, pp. 1–1, 04 2018. 50
- [115] C. Geng, S.-J. Huang, and S. Chen, “Recent advances in open set recognition: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2020. 50
- [116] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li, “Open set learning with counterfactual images,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 50
- [117] O. Gune, A. More, B. Banerjee, and S. Chaudhuri, “Generalized zero-shot learning using open set recognition,” in *BMVC*, p. 213, 2019. 50

## REFERENCES

---

- [118] Y. N. D. Hongyi Zhang, Moustapha Cisse and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” *International Conference on Learning Representations*, 2018. 55
- [119] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017. 59
- [120] J. Opitz and S. Burst, “Macro f1 and macro f1,” *ArXiv*, vol. abs/1911.03347, 2019.
- [121] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” in *ICML*, 2021. 86
- [122] W. Scheirer, A. Rocha, R. Micheals, and T. Boulton, “Meta-recognition: The theory and practice of recognition score analysis,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, 03 2011.
- [123] A. Rozsa, M. Günther, and T. E. Boulton, “Adversarial robustness: Softmax versus openmax,” 2017.
- [124] S. Prakhya, V. Venkataram, and J. Kalita, “Open set text classification using convolutional neural networks,” *International Conference on Natural Language Processing, 2017*, Dec 2017.
- [125] P. Perera, V. I. Morariu, R. Jain, V. Manjunatha, C. Wigington, V. Ordonez, and V. M. Patel, “Generative-discriminative feature representations for open-set recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

## REFERENCES

---

- [126] X. Sun, Z. Yang, C. Zhang, K.-V. Ling, and G. Peng, “Conditional gaussian distribution learning for open set recognition,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [127] P. Oza and V. M. Patel, “C2ae: Class conditioned auto-encoder for open-set recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [128] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, “Classification-reconstruction learning for open-set recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [129] S. Esmaeilpour, B. Liu, E. Robertson, and L. Shu, “Zero-shot open set detection by extending clip,” 2021. 88
- [130] Y. Atzmon and G. Chechik, “Adaptive confidence smoothing for generalized zero-shot learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [131] M. Bustreo, J. Cavazza, and V. Murino, “Enhancing visual embeddings through weakly supervised captioning for zero-shot learning,” in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.
- [132] H. Valpola, “From neural pca to deep unsupervised learning,” in *Advances in independent component analysis and learning machines*, pp. 143–171, Elsevier, 2015.
- [133] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

## REFERENCES

---

- [134] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
- [135] 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 *The Springer European Conference on Computer Vision (ECCV)*, 2016.
- [136] K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada, “Open set domain adaptation by backpropagation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [137] P. Panareda Busto and J. Gall, “Open set domain adaptation,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 754–763, 2017.
- [138] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, “Universal domain adaptation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2720–2729, 2019.
- [139] K.-C. Peng, Z. Wu, and J. Ernst, “Zero-shot deep domain adaptation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 764–781, 2018.
- [140] J. Wang and J. Jiang, “Adversarial learning for zero-shot domain adaptation,” in *European Conference on Computer Vision*, pp. 329–344, Springer, 2020.
- [141] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *ArXiv*, vol. abs/1710.10196, 2018.



# Appendix A

## Supplementary Material for Chapter 4

### A.1 Recent Advances in Visual-Language Models

Recently, deep visual and language models have been successfully combined for Zero-Shot transfer through a model named CLIP (Contrastive Language-Image Pre-training) [121]. Zero-Shot transfer aims at using a pretrained model with no fine-tune at all on new datasets. To perform Zero-Shot transfer, during the training, CLIP model learns to maximize the similarity between images and their corresponding textual description, as for example the description “*Pepper the aussie pup*” associated to a picture of an australian shepherd dog-breed puppy [121]. Since it does not use explicit labels for classification, at inference time a pre-computed set of descriptions can be created and their embeddings can be used for classification. For example, if, after training, we want to classify `cats` and `dogs`, we can:

## A.1 Recent Advances in Visual-Language Models

---

- construct two fixed descriptions “This is a picture of a cat” and “This is a picture of a dog”,
- compute the embeddings of the descriptions with the language model,
- compute the embedding of a new images with the visual model,
- classify the images accordingly with the similarity between the embedding of the image and the embeddings descriptions.

This is similar to what happens in Zero-Shot Learning when a visual features are compared to the class embeddings in the compatibility function methods presented in Section 2.5. The main differences is that with their pretraining it is not required by the model to learn the compatibility between a specific class embeddings and the visual features on the new dataset. In fact their pre-training already learns the compatibility between visual and textual features with the combination of visual and language models. Differently current Zero-Shot Learning methods rely on visual features extracted from a pure visual model, thus the compatibility has to be learned and it is achieved using the seen data.

Differently from Zero-Shot learning, this framework does not guarantee the Zero-Shot Learning assumption that the unseen classes have not been seen during training. In fact, CLIP model is based on a massive training using 400 million of images and text pairs collected from a variety of publicly available sources on the Internet and aims at Zero-Shot transfer on a new dataset thanks to a very general and extended pre-training.

However, the main idea of this paper is very valuable in Zero-Shot Learning and worth been investigated and improved. But the joint learning of textual descriptors and visual features can overcome the major limitations of the currently used families of class embeddings presented in Section 2.1.1.

## A.1 Recent Advances in Visual-Language Models

---

As we presented in the section it is difficult to obtain rich class embeddings that do not require large human effort in annotation and the idea of computing rich and inexpensive class embeddings is attractive and relevant for Zero-Shot Learning and computer vision community due to the possibility to generalize models to unseen categories.

A first study in producing class embeddings based on textual description for Zero-Shot Learning is proposed in [25] and presented in Section 2.1.1. However, while [25] requires very specific textual descriptions, thus difficult to obtain and limiting its applicability, the pre-training of CLIP is based on very raw and general description, increasing the computational cost, but allowing learning from a much wider source of information. In future studies, will be very important to balance raw and rich descriptions, in fact, as author point out in their work, CLIP model is very dependent on the availability and the quality of the text and images pairs and can struggle in fine-grained dataset as FLO [7], that is a standard benchmark in Zero-Shot Learning, and other fine-grained datasets. In particular authors point out that the choice and the construction of the textual embeddings is very important, named by authors as prompt engineering, and can help the model to achieve better performance. For example “A photo of a {label}” can be improved adding additional information as “a type of flower”

Very interestingly, in [129], authors propose to extend the Zero-Shot transfer problem to an Open Set framework, similarly to what we propose in this thesis for the Generalized Zero-Shot Learning problem. To address this problem, authors propose to extend CLIP to manage unknown categories. Since CLIP is trained to jointly learn visual and textual features, the authors use a text decoder on top of CLIP image embeddings to generate text. From this generated text, candidate unknown classes, that is words not belonging to the known ones, are extracted and new embeddings are created with the same CLIP prompt engineering procedure.

## A.2 Proposed Splits for Open Zero-Shot Learning

---

Thus for every image a set of possible unknown embeddings are created and used for classification, as well with the known ones. We differently, with VAcWGAN proposed to generate the unknown class embeddings in the latent space used for image generation.

## A.2 Proposed Splits for Open Zero-Shot Learning

In this pages, we provide the actual unseen and unknown classes that we considered for AWA [5], CUB [6], SUN [8] and FLO [7]. In the following tables, ✓ will denote class to be unseen for a given split (representing that the class embedding is disclosed) while ✗ denotes those classes for which the class embedding is not available while visual data are missing as well (*i.e.*, the unknown). For brevity, we omit from the following tables the list of seen classes (provided of both visual and semantic data) since this list is overlapping with the seen classes from the “Proposed Splits” of the survey [17]. We operated this choice to make our proposed Open Zero-Shot Learning setup complementary to the close-world setup of generalized zero-shot learning, so that practitioners can gradually shift towards the open-world regime - handling possibly many unknown classes while not forgetting neither seen nor unseen ones.

AWA [5]	
<i>Class Name</i>	<i>unseen</i>
horse	✓
blue+whale	✓
sheep	✗

## A.2 Proposed Splits for Open Zero-Shot Learning

---

seal	<i>x</i>
bat	<i>x</i>
giraffe	<i>x</i>
rat	✓
bobcat	✓
walrus	<i>x</i>
dolphin	✓

## A.2 Proposed Splits for Open Zero-Shot Learning

---

CUB [6]	
<i>Class Name</i>	<i>unseen</i>
004.Groove_billed_Ani	✓
012.Yellow_headed_Blackbird	✓
023.Brandt_Cormorant	✓
026.Bronzed_Cowbird	✓
028.Brown_Creeper	✓
031.Black_billed_Cuckoo	✗
033.Yellow_billed_Cuckoo	✗
043.Yellow_bellied_Flycatcher	✗
045.Northern_Fulmar	✓
049.Boat_tailed_Grackle	✗
052.Pied_billed_Grebe	✗
055.Evening_Grosbeak	✗
070.Green_Violetear	✓
072.Pomarine_Jaeger	✗
077.Tropical_Kingbird	✗
084.Red_legged_Kittiwake	✓
087.Mallard	✓
091.Mockingbird	✓
094.White_breasted_Nuthatch	✗
097.Orchard_Oriole	✓
098.Scott_Oriole	✗
103.Sayornis	✗
104.American_Pipit	✗

## A.2 Proposed Splits for Open Zero-Shot Learning

---

111.Loggerhead_Shrike	✗
113.Baird_Sparrow	✓
119.Field_Sparrow	✓
123.Henslow_Sparrow	✗
124.Le_Conte_Sparrow	✓
127.Savannah_Sparrow	✗
130.Tree_Sparrow	✓
132.White_crowned_Sparrow	✗
136.Barn_Swallow	✗
138.Tree_Swallow	✓
139.Scarlet_Tanager	✓
143.Caspian_Tern	✗
148.Green_tailed_Towhee	✗
156.White_eyed_Vireo	✗
157.Yellow_throated_Vireo	✓
161.Blue_winged_Warbler	✗
163.Cape_May_Warbler	✗
164.Cerulean_Warbler	✓
165.Chestnut_sided_Warbler	✗
168.Kentucky_Warbler	✗
169.Magnolia_Warbler	✓
173.Orange_crowned_Warbler	✓
180.Wilson_Warbler	✓
188.Pileated_Woodpecker	✗
190.Red_cockaded_Woodpecker	✓

## A.2 Proposed Splits for Open Zero-Shot Learning

---

191.Red_headed_Woodpecker	✓
200.Common_Yellowthroat	✓



## A.2 Proposed Splits for Open Zero-Shot Learning

SUN [8]	
<i>Class Name</i>	<i>unseen</i>
alley	<b>X</b>
archive	<b>X</b>
arena_basketball	✓
artists_loft	✓
auditorium	✓
ballroom	✓
bank_vault	✓
batting_cage_outdoor	✓
bazaar_indoor	<b>X</b>
bazaar_outdoor	<b>X</b>
betting_shop	✓
bog	<b>X</b>
bow_window_indoor	✓
bow_window_outdoor	✓
brewery_indoor	<b>X</b>
brewery_outdoor	<b>X</b>
bus_depot_outdoor	<b>X</b>
car_interior_frontseat	✓
casino_outdoor	<b>X</b>
chemistry_lab	✓
church_indoor	✓
church_outdoor	✓
doorway_indoor	<b>X</b>

## A.2 Proposed Splits for Open Zero-Shot Learning

---

elevator_interior	✓
excavation	✓
exhibition_hall	✗
field_cultivated	✓
firing_range_indoor	✗
fishpond	✓
galley	✓
geodesic_dome_indoor	✗
hangar_indoor	✗
hoodoo	✓
hotel_room	✗
ice_shelf	✗
jacuzzi_indoor	✗
japanese_garden	✓
lawn	✓
monastery_outdoor	✓
mosque_indoor	✓
motel	✗
observatory_outdoor	✓
parking_lot	✓
piano_store	✗
promenade_deck	✓
pub_indoor	✓
racecourse	✓
rectory	✗

## A.2 Proposed Splits for Open Zero-Shot Learning

---

sandbox	✓
savanna	✓
ski_resort	✓
temple_south_asia	✗
theater_indoor_seats	✓
ticket_booth	✓
trading_floor	✓
train_station_platform	✓
tundra	✗
tunnel_road_outdoor	✓
volleyball_court_outdoor	✓
workshop	✓
wrestling_ring_indoor	✗
yard	✗
ziggurat	✗

## A.2 Proposed Splits for Open Zero-Shot Learning

---

FLO [7]	
<i>Class Name</i>	<i>unseen</i>
Bird_of_paradise	<b>X</b>
Balloon_flower	<b>X</b>
Artichoke	✓
Alpine_sea_holly	✓
Barbeton_daisy	<b>X</b>
Bolero_deep_blue	✓
Buttercup	✓
Bishop_of_llandaff	✓
Black_eyed_susan	✓
Californian_poppy	✓
Bearded_iris	<b>X</b>
Azalea	✓
Anthurium	<b>X</b>
Bee_balm	<b>X</b>
Ball_moss	<b>X</b>
Bougainvillea	<b>X</b>
Camelia	✓
Bromelia	✓
Blanket_flower	<b>X</b>
Blackberry_lily	<b>X</b>