Zero-Shot Recognition of Generic Objects

Hascoet Tristan Erwan Marie

(Degree)
博士（学術）
(Date of Degree)
2019-09-25
(Date of Publication)
2020-09-01
(Resource Type)
doctoral thesis
(Report Number)
甲第7609号
(URL)
https://hdl.handle.net/20.500.14094/D1007609

博 士 論 文

Zero-Shot Recognition of Generic Objects
（ゼロショット学習を用いた一般物体認識）

２０１９年７月

神戸大学大学院システム情報学研究科

HASCOET Tristan Erwan Marie

Doctoral Thesis

Zero-Shot Recognition of Generic Objects

HASCOET Tristan Erwan Marie

Graduate School of System Informatics

Kobe University

July 2019

Doctoral Thesis

Zero-Shot Recognition of Generic Objects

HASCOET Tristan Erwan Marie

Image classification is a foundational task for computer vision. Within the past decade, Convolutional Neural Networks (CNN) have allowed for unprecedented progress in image classification. Since their early success in the ILSVRC classification challenge [1, 2], CNNs have become the backbone of modern computer vision as more complex vision systems, including semantic segmentation, object detection and visual question answering models, have been built on top of the original image classification architectures.

ZSL models generalize traditional image classifiers to recognize unknown classes, for which no image sample is available for training. The idea behind ZSL has largely been inspired by the human ability to define and recognize new object categories given a description: For example, a young child can recognize a zebra for the first time he sees one after his mother explains to him that a zebra looks like a horse with black and white stripes. Following this analogy, ZSL models use semantic representations of visual classes, i.e.; descriptions of the visual classes in a non-visual modality to transfer the discriminative knowledge learned from training classes to a set of unknown test classes.

The promise of ZSL for generic object recognition is huge: to scale up the recognition capacity of image classifiers beyond the set of annotated training classes. Hence ZSL has the potential to be of great practical impact as they would considerably ease the deployment of image classification models by eliminating the need for expensive task-specific data collection and fine-tuning processes. Furthermore, classification is a core building block of most vision systems including semantic segmentation and object detection models. As such, successfully generalizing classification to unknown classes would not only affect image classification systems, but also impact a wide array of computer vision systems relying on a classification functionality.

Despite its great promise, and after a decade of active research, the accuracy of ZSL models on the standard Imagenet dataset remains far too low to be considered for practical applications. In this thesis, we question several foundational elements of the current practice of ZSL for generic object recognition to shed some light on this apparent lack of progress:

We start by discussing the role of semantic representations in ZSL: we question the relevance and the limitations of word embeddings that have been widely adopted as the standard semantic representation for Zero-Shot Learning. We highlight several important limitations of defining large scale visual concepts with words and quantify the impact of these limitations on the accuracy of ZSL systems. We argue that instead of words, visual classes can be defined by explicit descriptions in the form of text documents and structured data, and propose to use Semantic Web technologies to automate the acquisition of such descriptions. We show that knowledge graph embeddings significantly improve the accuracy of existing ZSL models on the standard ImageNet ZSL benchmark.

We then question the current ZSL problem formulation: We argue that ZSL research should focus on the goal of combinatorial generalization of object recognition across classes. We show that the current standard benchmark is ill-defined in this regard, and introduce the notion of structural bias to back and quantify our claim. We show that structural bias in the standard benchmark has impacted the development of ZSL by favoring solutions based on a simple similarity search in semantic space and demonstrate that a trivial solution explicitly designed to take maximal advantage of structural bias outperforms almost all existing ZSL models. We conclude our analysis by proposing a new evaluation protocol explicitly designed to minimize the impact of structural bias.

In the final section, we investigate the application of invertible transformations to reduce the memory usage of large CNN. We analyze the memory bottleneck in training existing revertible networks and propose a layer-wise invertible architecture to alleviate these bottlenecks. We characterize the memory consumption and numerical errors arising in long chain of invertible transformations and, based on our analysis, propose an architecture with minimal memory footprint.

Together, we hope that the contributions presented in this thesis provide a solid foundation for future research into the Zero-Shot recognition of generic objects.

Contents

List of Figures

List of Tables

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other german or foreign examination board.

The related contents in this thesis have been previously published or submitted for publication by the author. A complete list of publications can be found on *pp.* xxi-xxii.

Publication List

Journal Papers

1. Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi: "Semantic embeddings of generic objects for zero-shot learning", *EURASIP Journal on Image and Video Processing*, 2019.1 (2019): 13.

2. Tristan Hascoet, Quentin Febvre, Yasuo Ariki, and Tetsuya Takiguchi: "Reversible designs for extreme memory cost reduction of CNN training", *EURASIP Journal on Image and Video Processing*, Under review.

3. Tristan Hascoet, Weihao Zhuang, Quentin Febvre, Yasuo Ariki, and Tetsuya Takiguchi: "Reducing the Memory Cost of Training Convolutional Neural Networks by CPU Offloading" *Journal of Software Engineering and Applications*, Under review, 12 pages.

International Conference Papers

1. Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi: "Semantic web and zero-shot learning of large scale visual classes", *Proceedings of the First WOrkshop on Symbolic Neural Learning, July 2017, Nagoya*

2. Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi: "Investigation of the correlations between CNN visual features and word embeddings", *Second Workshop on Closing the Loop Between Vision and Language, International Conference on Computer Vision, Deceber 2017, Venice.*

3. Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi: "Zero-shot learning using dictionary definitions", *International Workshop on Frontiers of Computer Vision. March 2018, Seoul.*

4. Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi: "On zero-shot recognition of generic objects", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019. p. 9553-9561.*

5. Tristan Hascoet, Weihao Zhuang, Quentin Febvre, Yasuo Ariki, and Tetsuya Takiguchi: "Extreme Memory Cost Reduction of CNN Training by Reversible Neural Network" I*nternational Conference on Computer Vision Workshop on Neural Architects*, October 2019, Seoul, Under review, 4 pages.

Domestic Conference Papers

1. Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi: "Visually grounded word embeddings for zero-shot learning of visual categories", *Computer Vision and Image Media, March 2018, Kyoto.*

2. Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi: "Knowledge graph embeddings for Zero-Shot Learning", *National Symposium on Image Processing and Understanding, August 2018, Sapporo.*

Glossary

ZSL	Zero-Shot Learning
CNN	Convolutional Neural Network
NLP	Natural Language Processing
GCN	Graph Convolution Network

Chapter 1

Introduction

1.1 Background

Image classification has been a foundational task of computer vision. Given an input image, image classifiers output a unique class label corresponding to the object depicted in the image. Image classification provides the problem with minimal functional definition to study the ability of algorithm to recognize objects from complex pixel patterns. For its minimal problem structure, image classification has been an important test bed for algorithmic development.

In the past decade, Convolutional Neural Networks (CNN) have allowed for unprecedented progress in image classification. Since their early success in the ILSVRC classification challenge [1, 2], CNNs have become the backbone of modern computer vision as more complex vision systems have been built on top of the original image classification architectures: Semantic segmentation models [6, 7, 8] extend the base CNN architecture of image classifiers with an expanding path to perform pixel-wise classification; object detection models [9, 10, 11] combine a classification module with region proposal and bounding box regression modules, and early image captioning models [12, 13] have successfully combined CNN classifiers with language models in an end-to-end architecture.

ZSL models extend traditional image classifiers to recognize unknown classes, for which no image sample is available for training. The idea behind ZSL has largely been inspired by the human ability to define and recognize new object categories given a description: For example, a children can recognize a zebra for the first time he sees one after his mother explains to him that a zebra looks like a horse with black and white stripes. Following this analogy, ZSL models use descriptions of visual classes, i.e.; representations of the visual classes in a non-visual modality to generalize the classification ability learned from training classes to a set of test classes.

1.1.1 Practical significance

Recent progress in computer vision has largely been driven by the successful application of Convolutional Neural Networks (CNN) trained in a supervised manner on large image datasets. One main drawback of these approaches is that they require a large amount of annotated data to successfully generalize to unseen image samples. The collection and annotation of such dataset for custom applications can be prohibitively complex and/or expensive, which hinders their applications to many real world practical scenarios. To reduce the sample complexity of CNN training, several research axis are being actively investigated, including unsupervised pretraining, semi-supervised learning, transfer learning, domain adaptation, and few-shot learning algorithms.

ZSL differs from these approaches, which rely at east on a few annotated samples of the target classes, as it represents the extreme case of few-shot learning in which classification among the target classes is performed without any training sample to learn from. The promise of ZSL for generic object recognition is huge: to scale up the recognition capacity of image classifiers beyond the set of annotated training classes. Hence ZSL has the potential to be of great practical impact as it would considerably ease the deployment of image classification models by eliminating the need for expensive task-specific data collection and fine-tuning processes.

Furthermore, classification is a core building block of most vision systems including semantic segmentation and object detection models. As such, successfully generalizing classification to unknown classes would not only affect image classification systems, but also impact down-stream systems relying on a classification module.

1.1.2 Theoretical significance

In addition to its potential practical impacts, ZSL provides an interesting setting for the investigation of several theoretical questions:

The first concerns the interaction between Computer Vision (CV) and Natural Language Processing (NLP): Recent years have seen an increasing amount of work sitting at the intersection of CV and NLP including image captioning, visual dialog and visual question answering tasks. State of the art models addressing these problems are typically made of vision and language models trained in an end-to-end manner, which hinders the interpretation of their processing and makes the investigation of their error cases difficult. Just as image classification has provided a minimal functional definition for the study of object recognition, ZSL provides a minimal problem structure to investigate the interactions between visual features and high-level abstractions as provided by feature representations from NLP. Hence, insights gained from ZSL research has the potential to benefit research at the intersection of CV and NLP.

The second, more conceptual, concerns the development of artificial intelligence: A key feature of human intelligence is the ability to make infinite use of finite means [14, 15]. For instance, natural languages allow for a small set of elements (words) to be composed in limitless ways (i.e.; new sentences). This reflects the principle of combinatorial generalization: constructing new inferences, predictions, and behaviors from known building blocks. Hence, combinatorial generalization has been considered a key challenge for the development of artificial intelligence [15]. In computer vision, object recognition is framed as a classification problem which explicitly defines a close world assumption: CV models can only recognize a finite set of image classes defined by the set of annotated sample available to learn from. This is in stark contrast with biological visual systems: humans can define and recognize a possibly infinite set of visual categories by combining known visual features (i.e. a zebra is a black and white striped horse-looking mammal). ZSL explicitly studies the ability to combine visual features learned from known classes to define and recognize unknown classes. Hence, ZSL provides the ideal setting to study the combinatorial generalization ability of object recognition models across visual classes. In chapter 3, we will argue that combinatorial generalization should be a defining goal of ZSL and propose a new evaluation protocol in this direction.

1.2 Approaches

ZSL models aim to recognize unseen classes, for which no image sample is available to learn from. To do so, ZSL models use descriptions of the visual classes, i.e., representations of the visual classes in a semantic space shared by both training and test classes. To evaluate the out-of-sample recognition ability of models, ZSL datasets split the full set of classes C into disjoint training and test sets. ZSL datasets are fully defined by three components: a set of training and test classes (C_{tr}, C_{te}), a set of labeled images X, and a set of semantic representations Y:

$$C_{tr} \cup C_{te} \subset C \tag{1.1a}$$
$$C_{tr} \cap C_{te} = \emptyset \tag{1.1b}$$
$$Y = \{y_c \in \mathbb{R}^d \quad \forall c \in C\} \tag{1.1c}$$
$$X = \{(x, c) \in \mathbb{R}^{3\times h\times w} \times C\} \tag{1.1d}$$
$$Tr = \{(x, y_c) \mid c \in C_{tr}\} \tag{1.1e}$$
$$Te = \{(x, y_c) \mid c \in C_{te}\} \tag{1.1f}$$

ZSL models are typically trained to minimize a loss function $\mathcal{L}$ over a similarity score E between image and semantic features of the training sample set with respect to the model parameters θ.

$$\theta^* = argmin_\theta \mathbb{E}_{(x,y)\in Tr} \mathcal{L}(E_\theta(x, y) + \Omega(\theta)) \tag{1.2}$$

In the standard ZSL setting, test samples x_{te} are classified among the set of unseen test classes by retrieving the class description y of highest similarity score:

$$c = argmax_{c\in C_{te}} E(x_{te}, y_c) \tag{1.3}$$

In the generalized ZSL setting, test samples are classified among the full set of training and test classes:

$$c = argmax_{c\in C} E(x_{te}, y_c) \tag{1.4}$$

The general architecture of ZSL models can be seen as the combination of three modules $\{V, S, E\}$, as illustrated in Figure 1.1. The visual module V extracts high-level visual features $V(x)$ from raw input images x; the semantic module S extracts semantic features $S(y)$ from raw descriptions y of the visual classes and the core ZSL module E computes a similarity score $sc = E(V(x), S(y))$ between semantic and visual features. In the following subsections, we describe different existing frameworks for the core ZSL module.

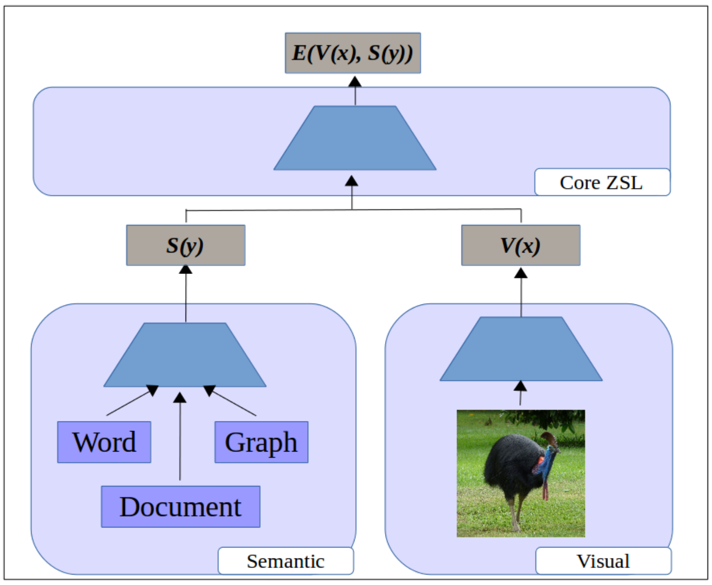

Figure 1.1: Illustration of ZSL model architecture

1.2.1 Regression

A number of existing models [16] frame ZSL as a regression problem. Given a visual features space $\mathcal{X}$ and semantic features $\mathcal{Y}$, such models learn a projection f_θ, parameterized by a set of parameters θ, from visual to semantic space so as to minimize a distance d between the semantic features and the visual feature projections from the set of training sample:

$$f_\theta : \mathcal{X} \rightarrow \mathcal{Y} \tag{1.5a}$$

$$\theta^* = argmin_\theta \mathbb{E}_{(x,y)\in Tr} d(f_\theta(x), y) \tag{1.5b}$$

At inference time, test images x are projected to the semantic space using the learned regression model. Classification is performed by selecting the semantic feature with minimal distance to the projected visual feature:

$$c^* = argmin_{c \in C_{te}} d(f_\theta(x), y_c) \tag{1.6}$$

1.2.2 Energy-based models

Energy-based models learn the unnormalized joint distribution of visual and semantic features. As such, energy-based models directly learn the similarity function E_θ, parameterized by a set of parameters θ, by maximizing the energy associated between matching pairs of visual and semantic features and minimizing it for non-matching pairs.

$$E_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \tag{1.7a}$$

$$\theta^* = argmax_\theta \mathbb{E}_{(x,y) \in Tr} E(x, y) - \lambda \mathbb{E}_{(x,y) \in \mathcal{X} \times \mathcal{Y} \setminus Tr} E(x, y) \tag{1.7b}$$

At inference time, test images x are classified into the visual class whose semantic representation yields maximal energy:

$$c^* = argmax_{c \in C_{te}} E_\theta(x, y_c) \tag{1.8}$$

1.2.3 Model averaging

Model averaging approaches compute similarity scores with the set of test classes as a weighted sum of a probability distribution over the training classes. The training step consists in training a regular classifier f_θ over the set of training classes.

$$f_\theta : x \rightarrow p(c|x), c \in C_{tr} \tag{1.9}$$

At inference time, given a test image x, and a weight matrix W, similarity scores with the test classes are computed as a weighted sum of the training class probabilities.

$$W = \mathbb{R}^{|C_{tr}| \times |C_{te}|} \tag{1.10a}$$

$$p(c_j|x) : x \rightarrow \sum_{i=0}^{|C_{tr}|} p(c_i|x) \times W_{i,j} \tag{1.10b}$$

1.3 Purpose and Novelties of This Thesis

Despite its great promise, and after a decade of active research, the accuracy of ZSL models on the standard Imagenet dataset remains far too low to be considered for practical applications.

The vast majority of concurrent research on ZSL focuses on the ZSL core module of the architecture presented in Figure 1.1, proposing different architectures, loss function, or regularization parameters. While these approaches have brought several incremental progresses on the standard benchmark, we believe that more fundamental key insights are missing to achieve significant improvement.

In this thesis, we question several foundational elements of the current practice of ZSL for generic object recognition:

We start by discussing the role of semantic representations in ZSL: we question the relevance and limitations of word embeddings that have been widely adopted as the standard semantic representation for Zero-Shot Learning. We highlight several important limitations of defining large scale visual concepts with words and quantify the impact of these limitations on the accuracy of ZSL systems. We argue that instead of words, visual classes can be defined by explicit descriptions in the form of text documents and structured data, and propose to use Semantic Web technologies to automate the acquisition of such descriptions. We show that knowledge graph embeddings significantly improve the accuracy of existing ZSL models on the standard ImageNet ZSL benchmark.

We then question the current ZSL problem formulation: We argue that ZSL research should focus on the goal of combinatorial generalization of object recognition across classes. We show that the current standard benchmark is ill-defined in this regard, and introduce the notion of structural bias to back and quantify our claim. We show that structural bias in the standard benchmark has impacted the development of ZSL by favoring solutions based on a simple similarity search in semantic space and demonstrate that a trivial solution explicitly designed to take maximal advantage of structural bias outperforms almost all existing ZSL models. We conclude our analysis by proposing a new evaluation protocol explicitly designed to minimize the impact of structural bias.

Finally, we investigate the application of invertible transformations to reduce the memory usage of large CNN. We analyze the memory bottleneck in training existing revertible networks and propose a layer-wise invertible architecture to alleviate these bottlenecks. We characterize the memory consumption and numerical errors arising in long chain of invertible transformations and, based on our analysis, propose an architecture with minimal memory footprint.

1.4 Outline

To structure our discussion, we analyze the different axes presented above in different chapters: Chapter 2 focuses on the semantic branch of the architecture depicted in Figure 1.1. Chapter 3 discusses the problem definition for the Zero-Shot recognition of generic object task, and the last chapter focuses on the visual feature extraction module.

Chapter 2

Semantic Feature Extraction

The related publications for this chapter are [17, 18, 19, 20, 21, 22].

2.1 Introduction

To recognize unseen classes, ZSL models use descriptions of the visual classes, i.e., representations of the visual classes in a non-visual modality. Research in ZSL has been driven by relatively small scale benchmarks [23, 24] for which human-annotated visual attributes are available as visual class descriptions. In the case of generic object recognition, however, manually annotating each and every possible visual class of interest with a set of visual attributes is impractical. Hence, generalizing the zero-shot learning approaches developed on such benchmarks to the more practical case of generic object recognition comes with the additional challenge of collecting suitable descriptions of the visual classes.

Finding such description presents two challenges: first, the collection of these descriptions must be automated so as to not require an expensive human annotation process. Second, the collected descriptions must be visually discriminative enough to enable the zero-shot recognition of generic objects. Word embeddings are learned in an unsupervised manner from large text corpora so that they can be collected in a large scale without human supervision. Furthermore, their successful application to a number of Natural Language Processing (NLP) tasks have shown that word embedding representations encode a number of desirable semantic features, which have been naturally assumed to generalize to vision tasks. For these desirable properties, word embeddings have become the standard visual class descriptions used by recent zero-shot generic object recognition models([3, 4, 25, 26, 27]).

In Section 2.4, we first question the current consensus on using word embeddings for zero-shot recognition of visual classes and discuss the relevance and limitations of the application of word embeddings to vision tasks. We then argue that generic objects can also be described by either text documents or knowledge graph data that satisfy our requirements: these descriptions both contain visually discriminative information and are automatically collectible from the web in a large scale, without requiring human intervention. In the remainder or this chapter, we refer to the descriptions of visual classes in different modalities as different *levels* of descriptions: either as word-, document-, or graph-level descriptions. We then present state of the art methods to learn feature representations from each level of descriptions and evaluate the visually discriminative power of these representations on the zero-shot generic object recognition task.

Our investigation highlights large differences in the ZSL accuracy of our baseline model using different embeddings. First, we find all word embedding models to not be equal: using GloVe [28] representations instead of the Word2vec [29] vectors used by previous works almost doubles the accuracy of our baseline model on the test split proposed by [3]. effectively outperforming state-of-the-art results obtained by more sophisticated ZSL models. However, we highlight several limitations to word embeddings that limit their applicability to specific classes.

Second, we find that different levels of descriptions are better suited to different application settings: graph embeddings tend to outperform word and document embeddings for datasets with strong structural bias as graph embeddings explicitly encode hierarchical relationships. In fact, we show that embedding a simple taxonomy of visual classes with a recently proposed model [30] slightly outperforms the best performing word embeddings. Embedding the full Wordnet knowledge graph doubles the accuracy of the best performing ZSL model using word2vec semantic features.

Finally, we find that simple document embedding schemes of large Wikipedia articles perform favorably compared to state-of-the art embeddings of smaller documents drawn from Wordnet definitions.

2.2 Related work

While the majority of works on zero-shot generic object recognition have used word embeddings as semantic features, some works have explored the use of different semantic features, which we present in this section. In [31], the authors use different linguistic resources to derive semantic similarity scores between classes, between classes and attributes, and to automatically mine attribute-classes correspondence. Similar to our work, they automate the acquisition of semantic data from knowledge bases, but they focus on deriving semantic similarity scores and part attributes while we evaluate graph embedding models. [32] uses visual class co-occurrence statistics to perform ZSL. Given a training set of multi-labeled images and similarity scores between known and unknown labels, they use the co-occurrence distribution of known labels to predict the occurrence of unknown labels in test images. Their multi-label classification setting differs from the ZSL setting in which input images are classified into a unique class. [33] questions the limits of using a single data point (word embedding vectors) as semantic representations of visual classes because this setting does not allow the representation of intra-class variance of semantic concepts. They used Gaussian distributions to model both semantic and visual feature distributions of the visual classes. More related to our work, [34] investigates different semantic representations for zero-shot action recognition. They compare different representations of documents and videos, while we investigate the application of word, document and knowledge graph embeddings to zero-shot recognition of generic objects. A series of works of [35, 36, 37] compares the zero-shot classification accuracy obtained with semantic representations derived from words, taxonomy and manual attribute annotations on fine-grain or small scale ZSL benchmarks. Our investigation differs in that we are concerned with the more practical task of generic object recognition and we investigate a broader class of semantic features.

2.3 Method

The general architecture of ZSL models can be seen as the combination of three modules $\{V, S, E\}$, as formalized in Chapter 1 and illustrated in Figure 1.1 .

The visual and semantic modules can either be learned jointly with the core ZSL module in an end-to-end procedure by back-propagation of the error signal from the core ZSL module to the two lower modules, or they can be learned independently on unsupervised or auxiliary supervised tasks (e.g., pretraining the visual module on the ILSVRC classification task and pretraining the semantic module as an unsupervised word embedding model).

This chapter focuses exclusively on the semantic module: we question what raw descriptions y and embedding module S provide semantic features $S(y)$ that are most visually discriminative so as to enable zero-shot recognition of generic objects. We restrict our study to embedding models S learned independently from other modules, without visual supervision from the ZSL module.

We use the top layer activations of a pretrained ResNet50 as visual feature representations $V(x)$. We investigate different embedding models S and raw semantic descriptions y in the form of words, text documents and knowledge graphs as semantic features $S(y)$. Our ZSL module consists of a ridge regression from the visual feature space to the semantic feature space:

Let us denote by (X, Y) the matrix representation of stacked visual and semantic features of the training set, we learn a projection matrix W from visual feature space to semantic feature space as:

$$W^* = min_W(||XW - Y||^2 + \lambda||W||^2) \tag{2.1a}$$

$$W = (XX^T + \lambda I)^{-1}XY^T \tag{2.1b}$$

At test time, similarity scores are given by the Euclidean distance between the projection of test images x in semantic space and the test class semantic features.

$$c^* = argmin_{c \in C_{test}}||V(x)W - S(y_c)|| \tag{2.2}$$

We use the simplest ZSL module possible for interpretability, to emphasize the importance of the semantic features, although we found qualitatively similar results with more sophisticated models. Previous works [38, 39] have shown that the Hubness problem negatively impact the ZSL accuracy of the ridge regression

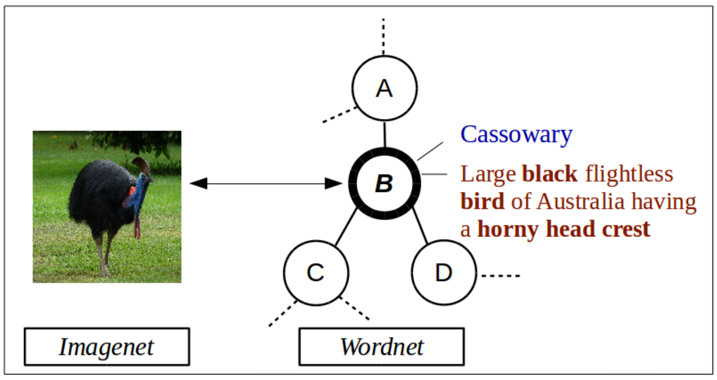
A
B
C
D
Cassowary
Large black flightless bird of Australia having a horny head crest
Imagenet
Wordnet

Figure 2.1: Illustration of the different description levels of generic object classes. ImageNet classes are indexed by Wordnet concepts which are defined by their lemmas (word level, in blue), definition (document level, in red) and structured data (graph level, in black)

model. We found this to be a non-problem as normalizing semantic features to unit norm solves the distance concentration in semantic space.

To conduct our study, we are heavily dependent on the data available to us in the form of image-description pairs (x, y). We use the ImageNet dataset as our starting point: In ImageNet, visual classes are indexed by Wordnet concepts, which are defined by three components that correspond to the three levels of representations we investigate: their lemmas (a set of synonym words that refer to the concept), a definition in natural language, and a node connected by a set of predicate edges to other concept nodes of the Wordnet knowledge graph. Figure 2.1 illustrates the different levels of descriptions provided by Wordnet.

In section 2.5, we will show how the semantic web can be used to automatically collect document- and graph-level descriptions of visual classes that are order of magnitudes larger than the descriptions provided by Wordnet. We first discuss the relevance and limitations of word embeddings in the following section.

2.4 Word Embeddings

2.4.1 Overview

Distributional Semantic Models (DSM) and neural word embeddings are two related classes of models that learn continuous distributed representations of words. These models implement the distributional hypothesis that states that the meaning of words can be defined by the context in which they occur. DSMs explicitly factorize matrices of word co-occurrence statistics while neural word embedding models learn word representations by stochastic optimization methods. The latter typically sample individual words and their context from large text corpora, maximizing a similarity score between co-occurring words. These approaches have been extensively studied both theoretically [40] and practically [41]. In [40], the authors show that the skip-gram word2vec model with negative sampling implicitly factorizes a shifted PMI matrix, suggesting that both approaches are qualitatively similar. For the sake of the following discussion, we consider that word embedding models do implicitly factorize matrices derived from word co-occurrence statistics following [40]. While qualitatively similar, the empirical study of [41] showed that neural embedding approaches tend to outperform DSM models on standard benchmarks. In Section 2.8, we evaluate three state-of-the-art embedding models on our ZSL benchmark: GloVe [28], FastText [42] and word2vec [29].

2.4.2 Relevance

Word embeddings have been shown to efficiently encode lexical and semantic similarities between words. Their successful application to NLP tasks has prompted word embeddings as standard semantic representations for zero-shot learning, while there has been little discussion as to the relevance and limitations of their application to vision task.

Slightly different from the original distributional hypothesis, ZSL models using word embeddings as semantic features make the assumption that the *appearance of generic objects* can be characterized by the context in which their lemmas occur. Table 2.1 shows the co-occurrence frequency of a few common visual class

Table 2.1: Lemmas co-occurrence with visually discriminative words

	car	truck	bird	cassowary
wheel	1.3×10^{-4}	1.6×10^{-4}	2.0×10^{-6}	0.0
drive	1.0×10^{-3}	1.0×10^{-3}	2.2×10^{-5}	2.4×10^{-3}
wings	4.0×10^{-6}	0.0	1.6×10^{-4}	0.0
beak	0.0	0.0	5.2×10^{-5}	0.0
occ.	4.7×10^{5}	7.8×10^{4}	1.4×10^{5}	7.9×10^{2}

Statistics presented in this table were gathered from the English Wikipedia corpus with a context window size of 5 words. Columns correspond to visual class lemmas and rows correspond to visually discriminative words. The last row shows the number of occurrence of the visual class lemmas in the corpus. Upper rows show the frequency of occurrence of visually discriminative words within the context of visual class lemmas. For example, the upper left value denotes $p(wheel|car)$.

lemmas with words that explicitly characterize visual attributes. This table shows that visual class lemmas tend to share high co-occurrence frequency with either their part attributes (i.e., both car and truck co-occur more frequently with wheel than bird and cassowary do) or action verbs that implicitly impacts the shape of these objects (i.e. both cars and trucks are "drivable" vehicles). This suggests that the co-occurrence patterns of words in large text corpora indeed contain information regarding distinctive visual features shared among classes. Hence, the latent space learned by the implicit factorization of such matrix embeds visually discriminative information so that word embeddings provide suitable semantic representations for zero-shot learning applications. A particular exception that stands out from Table 3.1 is the word cassowary, which we discuss in the following section.

2.4.3 Limitations

Consider describing a small set of naturally occurring generic objects such as the "car", "truck" and "bird" classes presented in Table 2.1. Coarse-grain visual classes seem to be well defined by such common words, for which rich co-occurrence statistics can be easily collected from large text corpora. However, humans can identify categories well beyond the limited scope of such coarse-grain visual classes. As one considers larger visual class sets of finer grain, several complications arise, which we describe in the following subsections.

2.4.3.1 N-grams.

Different from coarse grain concepts, fine-grain concepts are often not best described by single words but by composition of words (e.g. n-grams such as "polar bear" or "blue jeans" vs. their unigram parent class "bear" and "trousers"). We found that 54.2% of ImageNet class lemmas are not single words but n-grams. N-grams representations can be computed (e.g., by averaging of their individual words) but we question whether n-gram embeddings can be as visually discriminative as single word embeddings.

We split the ImageNet dataset into unigram and n-gram lemmas class sets. N-gram representations were computed as the mean of their individual word embeddings. As lemma scarcity might be correlated to their being n-gram or unigram, we only used classes whose lemmas appear between 1000 and 100,000 times within the English Wikipedia corpus to reduce the influence of lemma scarcity. Figure 2.2 shows the top-1 accuracy of each split, following the evaluation protocol described for the lemma scarcity evaluation. Surprisingly, ZSL accuracy does not seem to suffer from the "n-gram-ness" of visual class lemmas as n-gram lemmas even outperform single word lemmas by an average of 2%.

2.4.3.2 Word occurence frequency

We found that fine-grain visual classes that are correctly defined by a single word tend to be defined by rare words (e.g. the rare lemma "cassowary" vs. the common lemma "bird" of its parent class). As discussed in the previous section, word embeddings are learned from their co-occurrence statistics in large text corpora. While visual clues are embedded in words co-occurrence patterns, a considerable amount of noise (i.e. non visually discriminative information) stems from random word co-occurrences. The example of Cassowary is illustrated in Table 2.1. The word "cassowary" only occurs 792 times in the English Wikipedia corpus in which it does not co-occur once with the visual bird-like attributes "wings" or "beak". Instead, "cassowary" randomly co-occurs once with the word

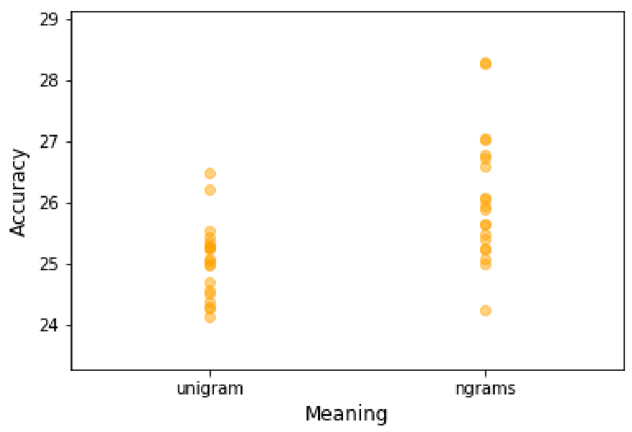

Figure 2.2: Evaluation of ZSL accuracy of n-gram classes vs. unigram classes. Left: n-gram vs. unigram class lemmas distribution. Right: classification accuracy per lemmas type.

"drive". We conjecture that frequently occurring words provide more visually discriminative representations than rare words because of the higher "visual signal to noise ratio" of their co-occurrence statistics. We found that 39% of ImageNet lemmas appear less than 100 times in Wikipedia.

Figure 2.3 shows the distribution of occurrence counts of the visual class lemmas in the full Wikipedia English corpus. To investiagate the impact of word occurence frequency on ZSL, we evaluate a baseline linear model on different test splits of 100 classes: we split the Imagenet classes into different subsets based on the occurrence frequency of their label word. We independently evaluate the accuracy of our model on each of these splits and report the ZSL accuracy with respect to the average occurrence frequency of the visual class labels.

Our results highlight a strong correlation ($r = 0.89$) between word frequency and ZSL accuracy as test splits made of rare words strikingly under-perform test splits made of more common words, although accuracy remains well above chance (1%), even for test sets of very rare words.

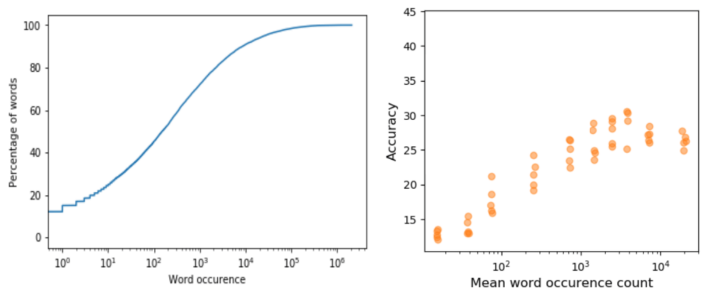

Figure 2.3: Evaluation of ZSL accuracy by test class lemmas occurrence frequency. Left: visual class lemmas occurrence count distribution in the English Wikipedia corpus. Right: classification accuracy per occurrence count.

2.4.3.3 Polysemy

Natural languages contain many polysemous words, which makes it difficult to uniquely identify visual classes with a single word. For example, a “(river) bank” and a “(financial) bank” share similar representations in a word embedding space while being two different visual concepts. The consequences of homonymy are two folds: first, the semantic representation of homonym classes is learned from the co-occurrence statistics of the different meanings of the lemma which results in noisy embeddings. Second, a mechanism to break ties between homonym visual classes must be given, which we describe below. We found that 13% of the ImageNet lemmas are shared with at least one other class and 38% of ImageNet classes share a lexical form with at least one other class.

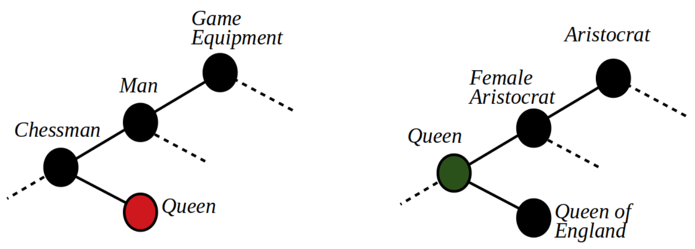

Figure 2.4: Illustration of two Wordnet concepts sharing the same label Queen.

Figure 2.4 gives an example of polysemous visual classes of the Imagenet dataset. To deal with polysemy, we want to assign a unique visual class to polysemous words. To do so, we define a similarity score $s(w, c)$ between words w and their visual classes c. Given a polysemous word w, we assign w to its visual class c of highest similarity score:

$$s : W \times C \to \mathbb{R} \tag{2.3a}$$

$$c^* = argmax_{c \in C} s(w, c) \tag{2.3b}$$

As a similarity score, we use the cosine similarity between word embeddings and the average word embedding of visual class parent and children concepts. Consider the example of the word *Queen* illustrated in Figure 2.4. There are 9 visual classes associated with the word *Queen* in the Imagenet dataset. For brievity, we only consider two of the *Queen* visual classes: one as an *Aristocrat*, and one as a *chesspiece* The similarity score between *Queen* and its *Aristocrat* visual class is given by:

$$s(c, w) = cos(w_{Queen}, \times(w_{Aristocrat} + w_{Female} + w_{England})/3) \tag{2.4a}$$

$$s(c, w) = 0.23 \tag{2.4b}$$

The similarity score between *Queen* and its *Chess* visual class is given by:

$$s(c, w) = cos(w_{Queen}, w_{Chessman}) \tag{2.5a}$$

$$s(c, w) = -0.04 \tag{2.5b}$$

So we assign the word *Queen* to the visual class of highest similarity score: The one corresponding to the *Aristocrat* meaning. Repeating this process for each polysemous word, we define a unique mapping between words and visual classes.

We conduct an experiment to assess both the impact of polysemy on ZSL accuracy and the efficiency of our solution. As in the previous section, we evaluate our ZSL models on different test splits of 100 classes: We separately evaluate test classes identified as the primary meaning of their word label and test classes corresponding to the secondary meaning of their word label. Figure 2.5 reports

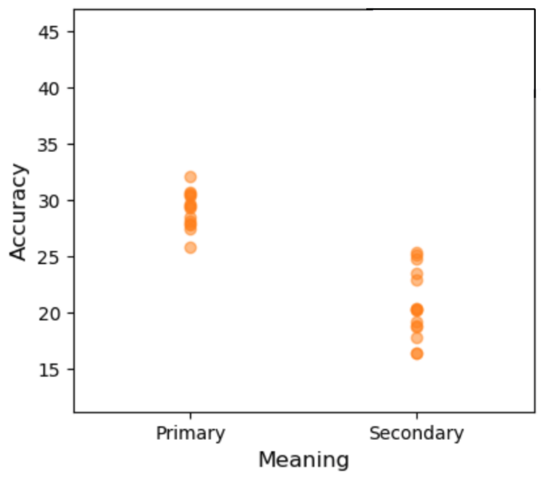

Figure 2.5: Evaluation of ZSL accuracy on primary vs. secondary meaning labels.

the accuracy obtained on these different test splits. We can see a significant boost in the ZSL accuracy of test classes whose word labels are identified as primary meanings. In comparison, test splits made exclusively of secondary meanings performed poorly. This confirms that polysemy does indeed impact ZSL accuracy, and suggests that our solution for primary meaning identification allows addressing this problem.

2.5 Data Augmentation

Table 2.2: Comparison of knowledge base statistics

	Documents		Graphs		
	doc	w/doc	nodes	edges	triples
Wordnet	117k	10	117k	20	372k
Babelnet	-	-	15M	2.3k	1.3G
Wikipedia	5.6M	630	-	-	-

(left) Number of document and average size (word per documents) of Wordnet definitions vs. Wikipedia articles. (right) number of nodes, edge types and triples of Wordnet vs. Babelnet knowledge graphs

The limitations of word-level representations highlighted in the previous section motivate us to investigate graph- and document-level descriptions. In this section, we focus on the automated acquisition of raw descriptions y to augment the Wordnet definitions and knowledge graph.

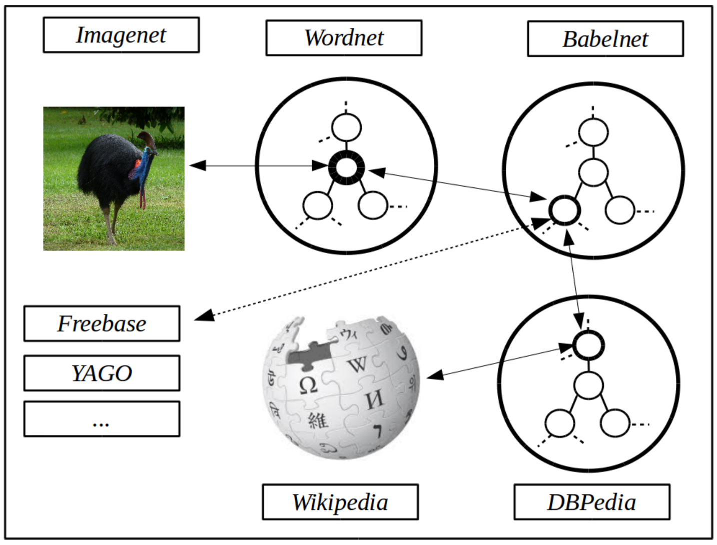

Figure 2.6: Illustration of our proposed linking process. ImageNet classes are indexed by Wordnet concepts. Wordnet concepts are linked to Babelnet concepts. Babelnet concepts are linked to DBPedia concepts. Each concept in DBPedia is linked to a Wikipedia article. Following links between Linked Open datasets allow us to collect rich descriptions of ImageNet visual classes. Our evaluation focused on Babelnet graph embeddings and Wikipedia articles embeddings but other knowledge bases such as Freebase or YAGO may be used interchangeably.

An interesting feature of WordNet is its integration to the Linked Open Data (LOD) cloud. Linked Data [43] refers to a set of best practices for publishers to integrate their data into a web of data; i.e, the semantic web. The LOD cloud references openly published datasets that follow the Linked Data best practices. Datasets of the LOD cloud contain links from their resources to resources of other LOD datasets. These links define equivalences between Wordnet concepts and resources of larger, richer knowledge bases. In particular, Wordnet concepts have been fully mapped to entities of the Babelnet [44] knowledge graph. Babelnet entities are also linked to external knowledge bases such as DBPedia or Freebase.

Following the links of the LOD cloud, as illustrated in Figure 2.6, we are able to collect descriptions of ImageNet classes orders of magnitude larger than the graph and document descriptions provided by Wordnet in a fully automated process. In our experiments, we used the Babelnet knowledge graph as augmented graph-level descriptions and Wikipedia articles as augmented document-level descriptions. Table 2.2 summarizes statistics of these datasets to illustrate the scale of this data augmentation.

2.6 Graph Embeddings

A knowledge graph can be formalized as a set of facts $\mathcal{G} = \{(s, p, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$. Each fact in the graph consists of a predicate (edge) $p \in \mathcal{R}$ and two entities (nodes) $s, o \in \mathcal{E} \times \mathcal{E}$, respectively referred to as the subject and object of the triple. Each triple denotes a relationship of type p between the subject s and the object o. Learning distributed representations of knowledge graph nodes and edges has been extensively studied in the framework of Statistical Relational Learning (SRL) [45]. SRL models are concerned with knowledge base completion tasks in which models aim to recover missing facts from large and incomplete knowledge graphs. In the following subsection, we review a subset of the literature on knowledge graph embeddings and, in Section 2.8, we evaluate the representations learned by these models on our ZSL benchmark. We refer the reader to [46] for more in-depth coverage of state-of-the-art models.

In addition to knowledge graph embeddings we consider methods from network embeddings; i.e., embeddings of non-typed edge graphs. Recently, two concurrent works ([47], [30]) have shown the benefits of hyperbolic space properties for embedding tree-like hierarchical data. In particular, [30] has shown impressively low reconstruction errors of hyperbolic embeddings of the Wordnet hierarchy. Follow-up work [48] have shown that the Wordnet hierarchy can be embedded perfectly (i.e. with zero reconstruction error) in a two-dimensional Poincarre disk. Because of their impressive success, we include the model introduced in [30] in our evaluation in section 2.8. We briefly present this work in Section 2.6. At the time of this writing, hyperbolic embedding models have not yet been extended to graph structures with typed edges like knowledge graphs.

Hence, we apply [30] to the Wordnet hierarchy (the subset of $\mathcal{G}$ considering only the Hypernym-Hyponym predicates).

2.6.1 Knowledge graph embeddings

Knowledge graph embedding models include tensor decomposition and neural embedding models. In [49], the authors show that simple neural embedding baselines such as DistMult [50] tend to outperform more sophisticated approaches on several benchmark knowledge base completion tasks, which leads us to focus on baseline neural embedding models. Neural embedding models learn d-dimensional vector representations of entities $\{e_i \in \mathbb{R}^d, \forall i \in \mathcal{E}\}$ and relations $\{r_i \in \mathbb{R}^d, \forall i \in \mathcal{R}\}$ by maximizing a scoring function $\psi(e_s, r_p, e_o)$ for triples $(s, p, o) \in \mathcal{G}$. Learning is performed stochastically by minimizing a loss function $\mathcal{L}$ over the score of randomly sampled triples:

$$e^*, r^* = argmin\Big(\mathbb{E}_{(s,p,o)\in\mathcal{G}}\mathcal{L}\left(\psi(e_s, r_p, e_o)\right)\Big) \tag{2.6}$$

Different embedding models differ in their choice of scoring function $\psi(e_s, r_p, e_o)$ and loss function $\mathcal{L}$ used for training. Table 2.3 summarizes the scoring function of popular models we evaluate in section 2.8.

Table 2.3: Neural embedding scoring and loss functions

Model	$\psi(e_s, r_p, e_o)$	$\mathcal{L}$
TransE [51]	$\|e_s + r_p - e_o\|$	L_2
DistMult [50]	$\langle e_s, r_p, e_o\rangle$	*Ranking*
ConvE [52]	$f(vec(f([e_s; r_p] * w))W)e_o$	*BCE*
TransE*	$\langle e_s + r_p, e_o\rangle$	*Triplet*

TransE [51] proposes to model relations r as translations in a Euclidean space, and considers the standard euclidean distance between translated subject and object embeddings as scoring function. DistMult [50] uses the trilinear dot product $\langle x, y, z\rangle = \sum_i x_i y_i z_i$ scoring function with a margin-based ranking loss function. ConvE [52] introduces depth in the scoring function. Their model consists of a convolution layer over the concatenation $[e_s, r_r]$ of the subject and predicate representations followed by a linear layer with ReLu activations. They use the dot product between the output of the network and the object embedding e_o as similarity score and the Binary Cross Entropy as loss function. While experimenting with these models, we found that representing relations as translations (similar to TransE) with a sigmoid dot product similarity score (similar to DistMult) yield highest accuracy. We also found the triplet margin loss to improve the quality of our embeddings. This variation is shown as $TransE^*$ in Table 2.3 and evaluated together with other baselines in the Experiment section.

2.6.2 Hyperbolic taxonomy embedding

The work of [30] proposed an original method called Poincarre embeddings to learn continuous embedding of symbols organized in a latent hierarchy. Instead of considering distances in a Euclidean space, the Poincarre model embeds symbols in a hyperbolic space in which the distance from data point u to v can be expressed as:

$$d(u, v) = arcosh\Big(1 + 2\frac{\|u - v\|^2}{(1 - \|u\|^2)(1 - \|v\|^2)}\Big) \tag{2.7}$$

Given a set of untyped edges $\mathcal{D} = \{(u, v)\}$, embeddings are learned by stochastic gradient descent so as to minimize the following loss function:

$$\mathcal{L}(\mathcal{D}) = \sum_{(u,v)\in\mathcal{D}} log\frac{e^{(-d(u,v))}}{\sum_{v'\in\mathcal{N}(u)} e^{-d(u,v')}} \tag{2.8}$$

Where $v' \in \mathcal{N}(u)$ represents a set of negative sample nodes v' that are not connected by an edge to u. The representations learned by this model have shown to better capture the hierarchical structure of taxonomies.

2.7 Document Embeddings

In this section, we review the literature for learning embeddings from text documents. Different models are concerned with documents of different scale, so we separately present embedding models for short, sentence-like documents (i.e. Wordnet definitions) and models concerned with full-text documents (i.e. Wikipedia articles). Sentence-level embeddings have been extensively studied for NLP applications, such as natural language inference or sentiment analysis, whereas full-text document embeddings have mainly been studied for information retrieval applications.

2.7.1 Sentence level embedding

The spectrum of different meanings that can be expressed by sentences is combinatorially more complex than the spectrum of meaning covered by individual words so that there is no agreed-upon universal sentence embedding model as there are for word embeddings. Instead, different models have been shown to be better suited to different NLP tasks.

Learning sentence embedding has been a very active topic of research over the last few years which has lead to a variety of architectures and training procedures. In this section, we discuss most relevant works, that we evaluate in Section 2.8.

Let $S = [w_0, w_1, ..., w_T]$ be a sentence of T words. Sentence embedding models produce a fixed-length representation $f(S)$ from variable-length sequences of words S. Different models differ in the choice of architecture and training signal used by the model.

2.7.1.1 Architectures

Most state-of-the-art models use either bag of words (BoW) or recurrent (Rec) architectures. BoW architectures represent sentences as the mean of transformations $g(w_i)$ applied to their individual words $w_i \in S$.

$$BoW(S) = \frac{1}{T} \sum_{i=1}^{T} g(w_i) \tag{2.9}$$

The BoW models we investigate use either the identity function $g(x) = x$ or linear projections $g_W(x) = Wx$. We respectively refer to these models as BoW_{id} and BoW_{lin}. Recurrent architectures sequentially process the words of a sentence by maintaining an internal state vector s_t at each step t of the processing, following equations (5). The exact formulation of functions g and f depends on the particular architecture used (LSTM, GRU or RNN).

$$s_t = g(s_{t-1}, w_t) \tag{2.10a}$$

$$o_t = f(s_t, w_t) \tag{2.10b}$$

$$Rec(S) = o_T = f(s_T, w_T) \tag{2.10c}$$

2.7.1.2 Training signals

Sentence embedding models can either be trained on supervised or unsupervised tasks. Supervised models are trained on corpora of labeled sentences, and use sentence labels as training signal. Unsupervised models either use the context information of neighboring sentences (inter-sentence objectives) in a corpus of ordered sentences as training signal or only use the information of their own words (intra-sentence objectives).

Supervised objectives. In [53], the authors argue that models trained on the task of natural language inference (NLI) learn universal representations of sentences that generalize well to other NLP tasks. Their model, InferSent, uses recurrent architectures trained on the MultiNLI dataset.

DictRep [54] is trained to map dictionary definitions of words to their word embeddings. They investigate both BoW_{lin} and $LSTM$ architectures.

Using similar architectures, the same work proposes CaptRep to learn visually grounded sentence representations by mapping image captions to image features. They use the MS-COCO dataset and extract visual features using a CNN.

Unsupervised inter-sentence objectives. The SkipThought model [55] uses a $LSTM$ sequence-to-sequence architecture. Given a corpus of ordered sentences $[S_0, S_1, ..., S_N]$, the model is trained to predict neighboring sentences S_{i-1}, S_{i+1} from a sentence S_i.

The FastSent model [56] uses adjacent sentences as prediction target similar to SkipThought. Unlike SkipThought, they use a BoW_{id} model with a log bilinear objective function to speed up the training process.

Unsupervised intra-sentence objectives. The Paragraph2vec model [57] extends the original word2vec model to sentences and document. To do so, [57] proposes to jointly learn sentence (or paragraph) representations as context words together with the word embeddings.

The Sent2vec [58] model proposes a different extension of the word2vec model: different from Paragrah2vec, the Sent2vec does not explicitly learn sentence representations. Instead, they model sentences with the BoW_{id} architecture. Sent2vec is trained to predict each individual word of a sentence given the BoW representation of the other words of the sentence.

Similar to SkipThought, Sequential Denoising Autoencoder (SDAE, [56]) uses a recurrent sequence-to-sequence architecture. Different from Skipthough, SDAE injects noise in the input sentences and uses the reconstruction loss of the output as training signal.

2.7.2 Full document embedding

Embedding full-length document has mainly been studied for document retrieval applications for which two of the most popular methods are Latent Semantic Indexing [59] and Latent Dirichlet Allocation [60]. Latent Semantic Indexing performs singular value decomposition on a matrix of term/document occurrence matrix. The TF-IDF model was introduced as a weighting factor to reduce the impact of frequently occurring words and has been shown to improve search results on document retrieval tasks. In section 2.8, we evaluate the TF-IDF representations of Wikipedia articles on our ZSL benchmark.

In addition to image retrieval, the processing of full-text documents has recently attracted the attention of the machine learning community for problems including abstractive text summarization [61] and open question answering [62]. These models typically use attention mechanisms to attend to specific segments of the document relevant to the task at hand. However, these models are not concerned with extracting fixed-length representations of the document content so that their integration to ZSL models is not straightforward. Integration of such models to the ZSL pipeline using the Wikipedia articles we provide represents one interesting direction for future research.

2.8 Experiments & Results

2.8.1 Methodology

In the following experiments, we used the 1,000 classes of the ILSVRC2012 image classification dataset as a training set. We use a ResNet-50 [63] as the visual module to extract 2048-dimension feature vectors from raw images. We use a ridge regression model as the core ZSL module. We evaluate the classification accuracy of our model using different semantic modules on different test splits as described in the following subsections.

2.8.2 Standard evaluation

Table 2.4 presents the results of our evaluation on standard test splits used in previous works [3, 4, 25, 26, 27].

General Observations. The bottom section of the table presents the state-of-the art results obtained using word2vec embeddings as reported in [4]. Note that the result they report for the ESZSL model using word2vec semantic vectors slightly underperform ours, which is could be due either to the different visual features we use or to the different preprocessing of the semantic features we performed.

Word embeddings. Table 2.4 highlights striking differences in performance between the word2vec embeddings used in previous works ([3, 4, 25, 26, 27]) and both GloVe and FastText embeddings. Using GloVe embeddings (13.47% top-1 accuracy on the hop-2 split), we are able to improve on the state-of-the-art (SYNC, 9.26%) by an absolute 4.21% on the top-1 accuracy of the hop-2 split.

Sentence embeddings. We expected sentence embedding to provide strong representations as they contain explicit references to fine-grain visual features of visual classes as illustrated by the definition of Cassowary given in Figure 2.1. Despite being a very active research topics, sentence embedding models performed surprisingly poorly on the hop-2 test split. We observe that models trained with supervised training signals tend to perform better than modes trained in an unsupervised manner. Among unsupervised models, models trained with intra-

Table 2.4: Results on the standard ImageNet ZSL test split proposed in [3].

			2-hop			3-hop			All		
ZSL module	description	semantic module	top-1	top-5	top-10	top-1	top-5	top-10	top-1	top-5	top-10
Ridge Reg.	Wordnet lemmas	*word2vec*	7.66	21.00	29.90	3.08	9.35	13.78	0.89	2.70	4.23
		FastText	12.98	32.35	41.68	2.96	9.01	13.20	1.30	4.00	6.01
		Glove	**13.47**	**32.96**	**42.99**	**3.08**	**9.35**	**13.78**	**1.34**	**4.15**	**6.30**
Ridge Reg.	Wordnet graph	*TransE*	5.77	8.73	10.16	1.07	1.71	1.99	0.42	0.65	0.76
		DistMult	16.94	37.61	43.85	3.28	9.57	12.39	1.35	3.91	5.08
		*TransE**	**20.13**	**48.32**	**58.06**	**3.65**	**11.84**	**17.05**	**1.51**	**4.90**	**7.21**
		ConvE	3.23	9.12	12.46	1.30	2.14	3.26	0.42	1.72	3.10
		Poincarre	11.81	28.86	37.53	2.02	5.93	8.79	0.79	2.32	3.46
Ridge Reg.	Babelnet graph	*TransE*	2.82	5.11	7.16	1.03	1.41	1.75	0.37	0.88	1.01
		DistMult	8.42	20.31	27.33	1.82	5.14	7.64	0.78	2.23	3.40
		*TransE**	17.76	42.46	53.47	3.62	10.82	15.65	1.53	4.69	6.97
Ridge Reg.	Wordnet definitions	*InferSent*	4.06	12.37	18.52	1.18	3.91	6.15	0.49	1.66	2.67
		DictRep	**6.06**	**18.74**	**27.28**	1.52	5.58	**9.05**	0.63	2.32	3.84
		CapRep	3.45	10.86	16.37	1.13	2.97	4.35	0.21	0.56	1.01
		Sent2vec	5.93	17.57	25.57	**1.65**	**5.60**	8.92	**0.67**	**2.36**	**3.85**
		FastSent	1.82	5.31	9.86	0.82	2.11	3.21	0.19	0.43	0.75
		SkipThought	0.50	1.38	2.11	0.17	0.46	0.67	0.06	0.17	0.26
Ridge Reg.	Wikipedia articles	*TFIDF*	9.03	26.53	37.31	-	-	-	-	-	-
			State-of-the-art								
SYNC [26]	Wordnet lemmas	*word2vec*	**9.26**	-	-	**2.29**	-	-	**0.96**	-	-
CONSE [27]			7.63			2.18			0.95		
ESZSL [16]			6.35			1.51			0.62		
ALE [37]			5.38			1.32			0.5		
LATEM [36]			5.45			1.32			0.5		
SJE [35]			5.31			1.33			0.52		
DEVISE[3]			5.25			1.29			0.49		
CMT [64]			2.88			0.67			0.29		
GCNZ [65]	Lemmas & graph	*Glove&GCN*	19.8	53.2	65.4	4.1	14.2	20.2	1.8	6.3	9.1
ADGPM [5]			**26.6**	**60.3**	**72.3**	**6.3**	**19.3**	**27.7**	**3.0**	**9.3**	**13.9**

The upper part of the table shows our results using a ridge regression model with different semantic representations. The bottom part of the table shows state-of-the-art results as reported in [4], with the additional entries of [5, 65].

sentence signals like Sent2vec seem to perform better than SkipThought or FastSent that use inter-sentence training signal.

Graph embeddings. Graph embeddings performed remarkably well on the hop-2 dataset. In particular, the TransE model with our proposed modifications outperformed the best performing word embedding model by an absolute 6.66% in top-1 accuracy (20.13% vs. 13.47%).

The Poincarre embeddings are learned from the Wordnet hierarchy alone so that they do not embed explicit visual information such as object part attributes.

It is remarkable that such embeddings performed on par with the best performing word embeddings.

ConvE embeddings performed poorly in contrast. This result seems to make sense: the ConvE model uses a non-linear similarity function which is not designed to learn linearly separable embeddings whereas we use a linear model as similarity measure between visual and semantic features.

Data augmentation. Wikipedia documents performed better than sentence embeddings which is encouraging. However, one major limitation of this approach is that many links between Wordnet concepts and Wikipedia articles could not be recovered from LOD data. Only 60% of visual classes were successfully matched to a Wikipedia article. Hence, we manually recovered the missing links for the hop-2 set (623 missing classes), but we did not recover the missing links for classes of larger test splits as manual linking is very time consuming.

Surprisingly, augmenting the Wordnet knowledge graph with Babelnet did not improve on the model's accuracy. The Babelnet knowledge graph is orders of magnitude larger than both Wordnet and standard SRL benchmarks against which knowledge graph embedding models are usually evaluated. This leads us to believe that models of greater capacity than the linear baselines we evaluated might be needed to extract more discriminative representations from such large knowledge bases.

2.9 Chapter Conclusion

In this chapter, we stressed out the importance of the semantic module on the accuracy of ZSL models. We discussed the relevance of word embeddings for ZSL and highlighted some limitations that lead us to believe that other levels of descriptions might be needed to achieve zero-shot recognition of generic objects.

We argued that, in addition to words, visual classes can be defined by text documents in natural language, which we referred to as document level descriptions, or by structured data as made available by large knowledge graphs, which we referred to as graph-level descriptions.

We showed that rich descriptions of visual concepts can be automatically scrapped from the web using linked open data and made this data openly available. We then reviewed the literature concerned with learning representations from such data and found that representations learned from explicit descriptions as made available by documents and knowledge graphs can indeed further outperform the best performing word embeddings on some zero-shot object recognition benchmark.

Architectures for the processing of graphs and documents are being actively researched. As research on graph and document embedding gains momentum, and given the limitations of word-level descriptions we outlined in Section 2.4, we expect semantic modules defined at the document and graph level to prove increasingly more efficient and we hope that the data we provide can prove useful for future research.

Chapter 3

Problem Definition

The related publications for this chapter are [66].

3.1 Introduction

Datasets play a leading role in computer vision research as they provide the framework on which algorithmic development is built. Perhaps the most striking example of the impact a dataset can have on research has been the introduction of Imagenet [2]. The new scale and granularity of Imagenet's coverage of the visual world has paved the way for the success and wide spread adoption of CNN [1, 67] that have revolutionized generic object recognition.

Despite its great promise, and after a decade of active research [68], the accuracy of ZSL models on the standard Imagenet benchmark [3] remain far too low for practical applications. In this chapter, we analyze the errors of several ZSL baselines on this benchmark to better understand this apparent lack of progress. Our analysis leads us to identify two main factors impacting the accuracy of ZSL models: structural flaws in the standard evaluation protocol and poor quality of both semantic and visual samples. On the bright side of things, we show that once these flaws are taken into account, the actual accuracy of existing ZSL models is much higher than was previously thought.

On the other hand, we show that a trivial solution outperforms most existing ZSL models by a large margin, which is upsetting. To explain this phenomenon, we introduce the notion of structural bias in ZSL datasets. We argue that the true promise of ZSL lies in the development of compositional reasoning abilities to achieve combinatorial generalization across classes. However, the presence of structural bias in the Imagenet benchmark favors solutions based on a trivial one to one mapping between training and test classes. We come to the conclusion that a new benchmark is needed to address the different problems identified by our analysis and, in the last section of this chapter, we detail the semi-automated construction of the new benchmark we propose.

To structure our discussion, we first briefly review related work in the next section. Section 3.3 details our analysis of the different factors impacting the accuracy of ZSL models on the standard benchmark. In Section 3.4, we introduce the notions of structural bias. We formalize this notion by proposing a measure to quantify the amount of structural bias for any training and test split drawn from the Imagenet dataset. Finally, Section 3.5 summarizes the construction of our proposed benchmark. This benchmark is explicitly designed to minimize structural bias and avoid the pitfalls of existing datasets.

3.2 Related Work

3.2.1 ZSL datasets

Early research on ZSL has been carried out on relatively small scale or domain specific benchmarks [24, 69, 70], for which human-annotated visual attributes are proposed as semantic representations of the visual classes. On the one hand, these benchmarks have provided a controlled setup for the development of theoretical models and the accurate tracking of ZSL progress. On the other hand, it is unclear whether approaches developed on such datasets would generalize to the more practical setting of zero-shot recognition of generic objects. For instance, in generic object recognition, manually annotating each and every possible visual class of interest with a set of visual attributes is impractical due to the diversity and complexity of the visual world. Hence, generalizing the zero-shot learning approaches developed on such benchmarks to the more practical case of generic object recognition comes with the additional challenge of collecting suitable descriptions of the visual classes.

The Imagenet dataset [2] consists of more than 13 million images scattered among 21,845 visual classes. Imagenet relies on Wordnet [71] to structure its classes: each visual class in Imagenet corresponds to a concept in Wordnet. Frome *et al.*[3] proposed a benchmark for ZS generic object recognition based on the Imagenet dataset, which has been widely adopted as the standard evaluation benchmark by recent works [4, 5, 16, 26, 27, 36, 65]. Using word embeddings as semantic representations, they use the 1000 classes of the ILSVRC dataset as training classes and propose different test splits drawn from the remaining 20,845 classes of the Imagenet dataset based on their distance to the training classes within the Wordnet hierarchy: the *2-hops*, *3-hops* and *all* test splits.

Careful inspection of these test splits revealed a confusion in their name: The *2-hops* test split actually consists of the set of 1589 test classes directly connected to the training set classes in Wordnet, i.e; within *1 hop* of the training set. Similarly, the *3-hops* test set actually corresponds to the test classes within *2-hops*. In this paper, we will refer to the standard test splits by the name of their true configuration: *1-hop*, *2-hops* and *all*, as illustrated in Figure 3.1.

3.2.2 Dataset bias

Bias in datasets can take many forms, depending on the specific target task. Torralba *et al.*[72] investigates bias in generic object recognition. The notion of structural bias we introduce in Section 3.5 is closely related to the notion of negative set bias they analyze.

As more complex tasks are being considered, more insidious forms of bias sneak into our datasets. In VQA, the impressive results of early baseline models have later been shown to be largely due to statistical biases in the question/answers pairs [73, 74, 75]. Similar to these works, we will show that a trivial solution leveraging structural bias in the Imagenet ZSL benchmark outperforms early ZSL baselines.

Xian *et al.*[4] identify structural incoherences in small-scale ZSL benchmarks and proposes new test splits to remedy them. Closely related to our work, they also observe a correlation between test class sample population and classification accuracy in the Imagenet ZSL benchmark. However, their analysis mainly focuses on small-scale benchmarks and the comparison of existing ZSL models, while we analyze the ZSL benchmark for generic object recognition in more depth.

3.3 Error analysis

In the introduction of this thesis, we have mentioned that ZSL benchmarks are fully defined by three components: a set of labeled images X, a set of semantic representations Y, and the set of training and test classes (C_{tr}, C_{te}). In this section, we analyze each of the standard benchmark components individually: We first highlight inconsistencies in the configuration of the different test splits and show that these inconsistencies lead to many false negatives in the reported evaluation of ZSL models outputs. Next, we identify a number of factors impacting the quality of the word embeddings of visual classes and argue that visual classes with poor semantic representations should be excluded from ZSL benchmarks. We then observe that the Imagenet dataset contains many ambiguous image samples. We define what a *good* image sample means in the context of ZSL and propose a method to automatically select such images.

3.3.1 Structural flaws

Figure 3.1 illustrates the configuration of test classes of the standard test splits within the Wordnet hierarchy. This configuration leads to an obvious contradiction: test sets include visual classes of both parents and their children concepts. Consider the problem of classifying images of birds within the *hop-1* test split as in Figure 3.1. The standard test splits give rise to two possibly inconsistent scenarios:

A ZSL model may classify an image of the children class *Cathartid* as its parent class *Raptor*. The standard benchmark considers such cases as classification errors, while the classification is semantically correct.

A ZSL model may classify an image of the parent class *Raptor* as one of its children class: *Cathartid*. Classification may be semantically correct or incorrect, depending on the specific breed of raptor in the image, but we have no way to automatically assess it without additional annotation. The standard benchmark

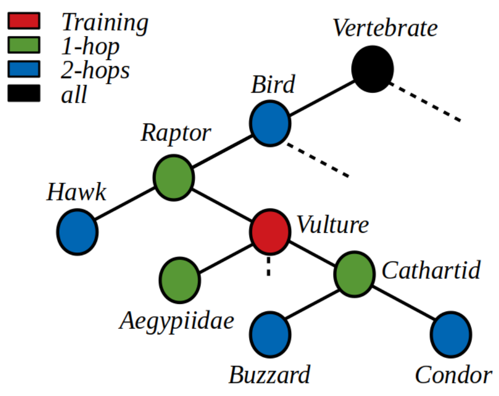

Figure 3.1: Illustration of the standard test splits configuration

considers such cases as classification errors, while the classification is semantically undefined.

We refer to both of the above cases as false negatives. Figure 3.2 illustrates the distribution of ZSL classification outputs among these different scenarios on the 1-hop test split. On the standard ZSL task for instance, the reported accuracy of the GCN model is 21.8% while the actual (semantically correct) accuracy should be somewhere in between 27.8% and 40.4%.

The ratio of false negatives per accuracy increases dramatically in the generalized ZSL setting. The linear baseline reported accuracy is only 1.9%, while the actual (semantically correct) accuracy lies between 16.0% and 41.1%. This is due to the fact that ZSL models tend to classify test images into their parent or children training class: for example, *Cathartid* images tend to be classified as *Vulture*.

Figure 3.3 and 3.4 illustrate the distribution of ZSL classification outputs on the *2-hops* and *all* test splits respectively. On the *2-hops* standard ZSL test set, 3.6% of test images were correctly classified by the Linear baseline model.

Table 3.1 summarizes the ratio of false negative per true positive on each of the standard test split: $ratio = FN/TP$. This table shows two interesting trends: First, the ratio is much higher in the Generalized ZSL setting due to the fact that ZSL models tend to classify test images as their parent or children training class. Second, in the standard ZSL setting, the ratio tends to increase with larger test

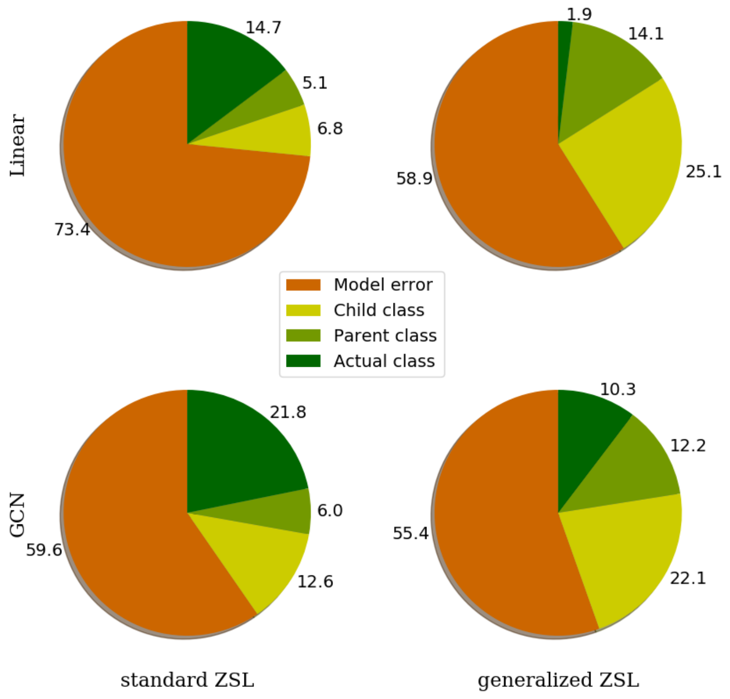

Figure 3.2: Distribution of the classification outputs of different ZSL models on the *1-hop* test split. An image x can be either be classified into its actual label c, the parent class of c, one of its children class, or an unrelated class. Only the latter case constitutes a definitive error.

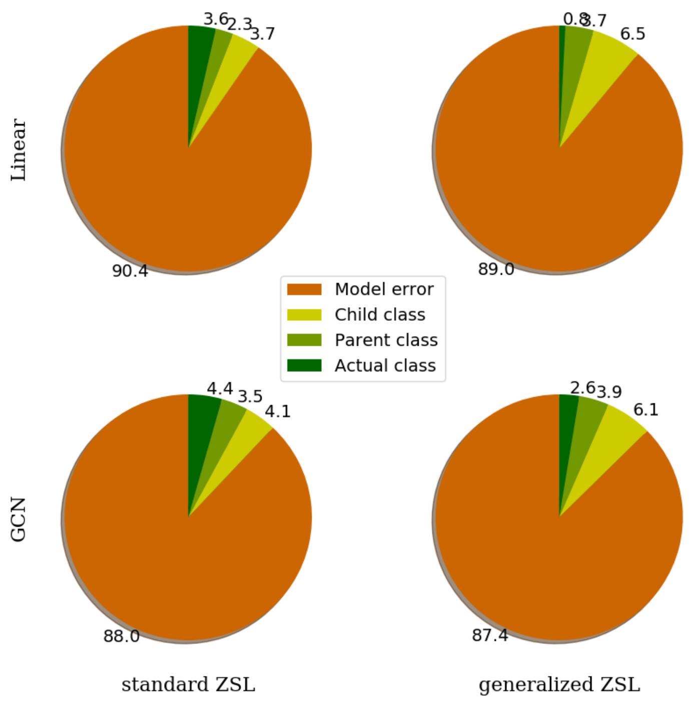

Figure 3.3: Distribution of classification outputs on the *2-hops* test split.

sets: the GCN model ratios are 2.3, 3.8 and 4.1 on the *1-hop*, *2-hops* and *all* test splits respectively. We believe this is due to larger overlaps within the Wordnet hierarchy: In the *1-hop* test set, the only FN classes for *Cathartid* images is *Raptor*. In the *2-hops* test set, *Buzzard*, *Condor*, *Raptor* and *Bird* are all FN classification outputs for *Cathartid* images. This trend, however, does not hold for the Linear model in the Generalized ZSL setting.

3.3.2 Word embeddings

In Chapter 2, we highlighted two major limitations of word embeddings: scarce occurence and polysemy. These problems naturally arise in the definition of large scale object categories so they are inherent problems of ZS recognition

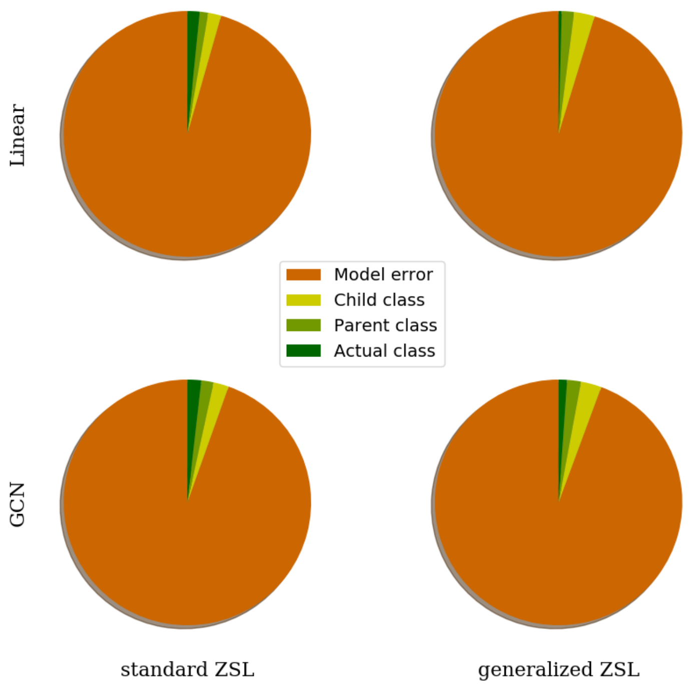

Figure 3.4: Distribution of classification outputs on the *all* test split.

Table 3.1: Ratio of false negatives (FN) per true positives (TP).

		1-hop			2-hops			all		
Model	Task	TP	FN	ratio	TP	FN.	ratio	TP	FN	ratio
Linear	ZSL	14.7	10.2	**0.7**	3.6	6.0	**1.7**	1.6	2.8	**1.7**
	GZSL	1.9	39.2	**20.6**	0.8	10.23	**12.7**	0.4	4.27	**10.7**
GCN	ZSL	21.8	18.6	**0.8**	4.4	7.6	**1.7**	1.8	3.6	**2.0**
	GZSL	10.3	34.2	**2.3**	2.6	10.0	**3.8**	1.1	4.5	**4.1**

of generic objects. However, we argue that ZSL benchmarks should provide a curated environment with high quality, unambiguous, semantic representations and that solutions to tackle the special case of polysemous and rare words should be separately investigated in the future.

Figure 3.5 summarizes the impact of both factors on the accuracy of our baseline linear model and a state-of-the-art GCN model. First, results highlight a strong correlation ($r = 0.89$) between word frequency and ZSL accuracy as test splits made of rare words strikingly under-perform test splits made of more common words, although accuracy remains well above chance (1%), even for test sets of very rare words. Results are more nuanced for the GCN model (correlation coefficient $r = 0.74$), which can be explained by the fact that GCN uses the Wordnet hierarchy information in addition to word embeddings. Second, we can see a significant boost in the ZSL accuracy of test classes whose word labels are identified as primary meanings. In comparison, test splits made exclusively of secondary meanings performed poorly. This suggests that the solution for primary meaning identification laid out in Chapter 2 addresses the problem of polysemy.

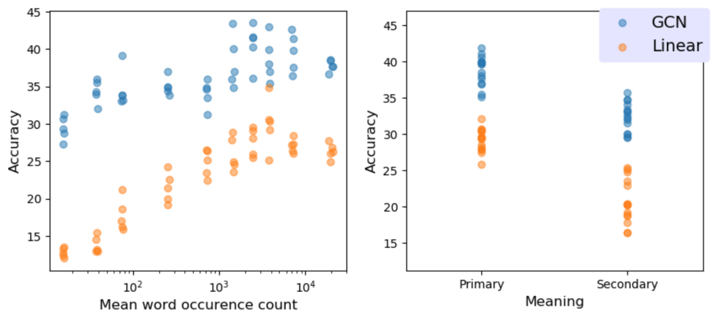

Figure 3.5: Impact of lemma scarcity and polysemy on ZSL accuracy.

Based on these results, we will rely on our primary meaning identification solution for the construction of a new dataset in Section 3.5. and only select classes whose word label are common enough to provide strong semantic representations.

3.3.3 Image samples

The ILSVRC dataset consists of a high-quality curated subset of the Imagenet dataset. The current ZSL benchmark uses ILSVRC classes as training classes and classes drawn from the remainder of the Imagenet dataset as test sets, assuming similar standards of quality from these test classes. Upon closer inspection, we found these test classes to contain many inconsistencies and ambiguities. In this section, we detail a solution to automatically filter out ambiguous samples so as to only select quality samples for our proposed benchmark.

3.3.3.1 Class-wise selection

Xian *et al.*[36] have first identified a correlation between the sample population of visual classes and their classification accuracy. They conjecture that small population classes are harder to classify because they correspond to fine-grained visual concepts, while large population classes correspond to easier, coarse-grained concepts. Manual inspection of these classes lead us to a different interpretation:

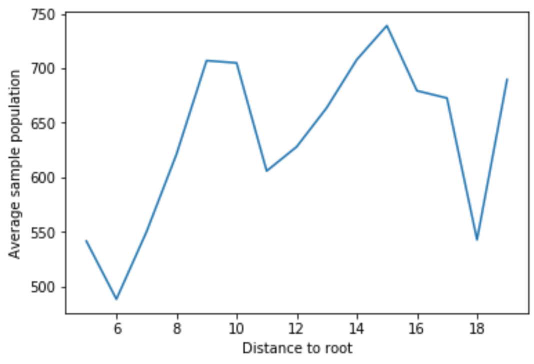

Figure 3.6: Average sample population per visual class with respect to their "granularity".

First, we found no significant correlation between sample population and concept granularity: Figure 3.6 shows the average sample population of visual classes with respect to their distance to the root node in the Wordnet hierarchy. We propose to use the distance of visual classes to the root node within the Wordnet hierarchys a measure of their "granularity" as fine-grain classes are lower in the Wordnet hierarchy, hence further away from the root node than coarse-grain classes. This figure illustrates no clear correlation between visual class granularity and their sample population. For instance, visual classes within 6 hops of the root node have an average sample population of 490 images, while visual classes within 10 hops of the root node have an average sample population of 700 images.

On the other hand, we found many visually ambiguous concepts such as ”ringer”, ”covering” or ”chair of state” to have low sample populations. Such visually ambiguous concepts are harder for crowd-sourced annotators to reach consensus on labeling, resulting in lower population counts. Hence, the sample population of visual classes seems to offer a proxy value to the ambiguity of visual classes rather than their granularity.

In Figure 3.7, we report the ZSL accuracy of our models on different test splits with respect to their average population counts. This figure shows a clear correlation between the sample population and the accuracy of both models, with low accuracy for low sample population classes. We use the sample population as a rough indicator to quickly filter out ambiguous visual classes and only consider classes with sample population superior to 300 images as valid candidate classes in our proposed dataset.

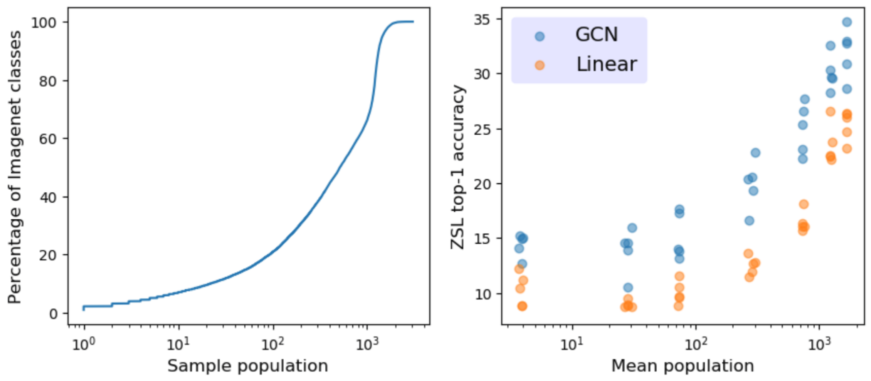

Figure 3.7: ZSL accuracy with respect to sample population sizes. Left: Distribution of Imagenet class population size. 6.1% of Imagenet classes have less than 10 samples, 21.1% have less than 100 samples. Right: ZSL accuracy of different test splits with respect to their mean sample population size.

3.3.3.2 Sample-wise selection

Even among the selected classes, we found many inconsistent and ambiguous images to remain (Appendix C), so we would like to further filter quality test images sample-wise. But what makes a good candidate image for a ZSL benchmark? How can we measure the quality of a sample? We argue that ZSL benchmarks should only reflect the *zero-shot ability* of models: ZSL benchmarks should evaluate the accuracy of ZSL models *relatively to the accuracy of standard non-ZSL models*. Hence, we define a good ZSL sample as an image unambiguous enough to be correctly classified by standard image classifiers trained in a supervised manner.

To automatically filter such quality samples, we fine-tune and evaluate a standard CNN in a supervised manner on the set of candidate test classes. We consider consistently miss-classified samples to be too ambiguous for ZSL and only select samples that were correctly classified by the CNN Specifically, we define our filtering procedure as follows:

Given a set of labeled samples $X = \{(x, c)\}$, our procedure returns a subset $X' \subset X$ of high-quality images. This selection process is formalized in Algorithm 1, and proceeds as follows:

First, we randomly sample subsets of 1000 visual classes $C' \subset C$ from the full Imagenet dataset. Classes are sampled so as to contain no overlap in the Wordnet hierarchy: random splits C' do not contain both parent and their children classes.

Second, we randomly sample 250 images per class as training samples, and use the remaining images as test samples. We fine-tune the last layer of a pretrained Resent-50 on the set of training samples, and evaluate the classification output of the model on the test samples.

We consider correctly classified image samples as high-quality test samples for our benchmark and discard the incorrectly classified images. We repeat this operation until all samples $x \in X$ have been evaluated. The output X' of this procedure is a subset of high-quality image samples that were correctly classified by the model.

3.3.4 Dataset Summary

Figure 3.8 summarizes the impact of the different factors we analyzed on the top-1 classification error of both our baseline models on the "1-hop" test split. The error rate of the Linear model on the standard ZSL setting drops from 86% to 61% after removing ambiguous images, semantic samples, and structural flaws.

```
Input:
Imagenet Dataset: $X = \{(x, c) \in \mathbb{R}^{3\times h\times w} \times C\}$
ILSVRC-pretrained ResNet: $BaseModel : \mathbb{R}^{3\times h\times w} \to C$
Output:
High-quality Imagenet subset: $X' \subset X$
Init:
Initialize an empty error set $Err = \emptyset$ and accurate set: $Acc = \emptyset$
while $Err \cup Acc \neq X$ do
    $C' = SampleClass(C, 1000)$
    $X_{C'} = \{(x, c) | c \in C'\}$
    $X_{train}, X_{test} = SampleSplit(X_{C'}, 250)$
    $Model = FineTune(BaseModel, X_{train})$
    for $(x, c) \in X_{test}$ do
        if $Model(x) == c$ then
            $Acc = Acc \cup \{(x, c)\}$
        else
            $Err = Err \cup \{(x, c)\}$
        end
    end
end
$X' = Acc$
end
```

Algorithm 1: Sample-wise selection procedure. $SampleSplit(C, n)$ is a sampling procedure that returns a subset C' of n non-overlapping classes (i.e.; no children classes and their parents are contained in C') from the class set C. $SampleSplit(X, n)$ is a sampling procedure that returns a training set X_{train} of n training samples for each class in X, and the remaining samples as a test set X_{test}. $FineTune(M, X)$ is a procedure that fine-tunes a model M on the input training set X.

The error rate of the GCN model on the generalized ZSL setting drops from 90% to 47%.

The GCN model is particularly sensitive to the structural flaws of the standard benchmark, but less sensitive to noisy word embeddings than the linear baseline.

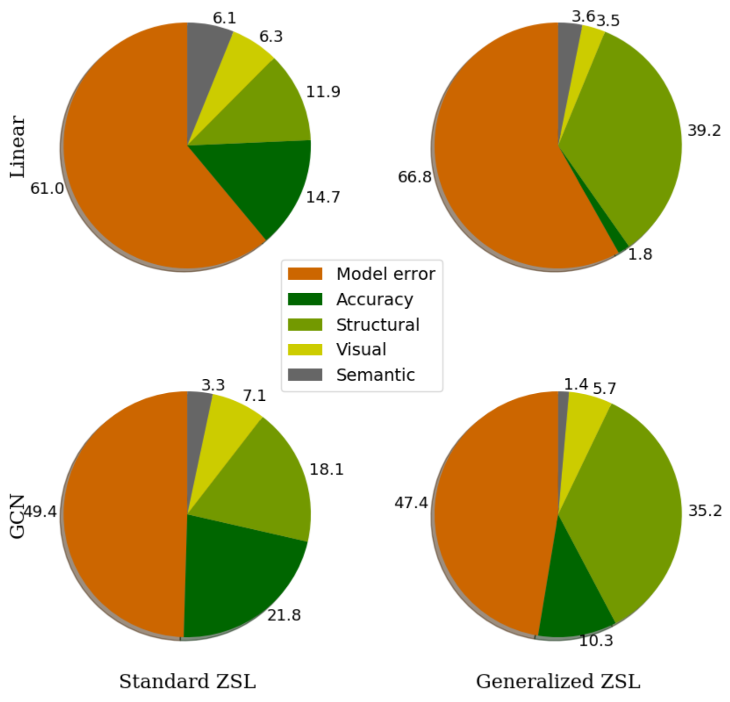

Figure 3.8: Estimation of the impact of different factors on the reported error of existing models on the 1-hop test split

This can be easily explained by the fact that GCN models rely on the explicit Wordnet hierarchy information as semantic data in addition to word embeddings.

3.4 Structural bias

ZSL models are inspired by the human ability to recognize unknown objects from a mere description, as it is often illustrated by the following example: Without having ever seen a zebra, a person would be able to recognize one, knowing that zebras look like horses covered in black and white stripes. This example illustrates the human capacity to *compose* visual features of *different* known objects to define and recognize previously unknown object categories.

Standard image classifiers encode class labels as *local representations* (one-hot embeddings), in which each dimension represents a different visual class, as illustrated in Figure 3.10 As such, no information is shared among classes in the label space: visual class embeddings are equally distant and orthogonal to each other. The main idea behind ZSL models is to instead embed visual classes into *distributed representations*: In label space, visual classes are defined by multiple visual features (horse-ish shape, stripes, colors) shared among classes. Distributed representations allow to define and recognize unknown classes by *composition* of visual features shared with known classes, in a similar manner as the human ability described above.

The embedding of visual classes into distributed feature representations is especially powerful since it allows to define a combinatorial number of test classes by composition of a possibly small set of features learned from a given set of training classes. Hence, we argue that the key challenge behind ZSL is to achieve ZS recognition of unknown classes by composition of known visual features, following their original inspiration of the human ability, and as made possible by distributed feature representations. In this section, we will see that not all ZSL problems require such kind of compositional ability. On the standard benchmark, we show that a trivial solution based on local representations of visual classes outperform existing approaches based on word embeddings. We show that this trivial solution is made possible by the specific configuration of the standard test splits and introduce the notion of structural bias to refer to the existence of such trivial solutions in ZSL datasets.

3.4.1 Toy example

Figure 3.9 illustrates a toy ZSL problem in which, given a training set of *Horse* and *TV monitor* images, the goal is to classify images of *Zebra* and *PC laptop*. Let's consider training an image classifier on the training set and directly applying it to images from the test set. We can safely assume that most zebra images will be classified as horses, and most laptop samples as TV monitors. Hence, a trivial solution to this problem consists in defining a one to one mapping between test classes and their closest training class: *Horse=Zebra* and *TV monitor=PC laptop*.

This example makes it fairly obvious that not all ZSL problems require the ability to compose visual features to solve.

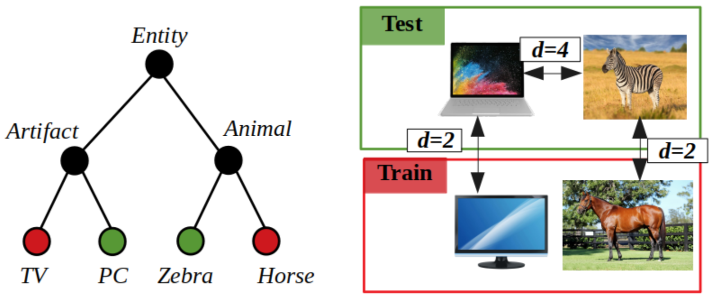

Figure 3.9: Illustration of the toy example. Left: Wordnet-like class hierarchy. Training classes are shown in red and test class in green. Right: Illustration of image samples. The black captions represent the distance between classes as their shortest path length.

Classification problems define a close-world assumption: As all test samples are known to belong to one of the test classes, classifying an image x into a given test class c means that x is more likely to belong to c than other classes of the test set. In other words, classification is performed relatively to a negative set of classes [72].

What made this trivial ZSL solution possible is the fact that test classes of our toy example are very similar to one of the training class, relatively to their negative set. This allowed us to identify a one-to-one mapping by similarity between training and test classes. We refer to this trivial solution as a similarity-based solution, in opposition to solutions based on the composition of visual features.

As illustrated in Figure 3.10, the similarity mapping between test and training classes can be directly embedded in the semantic space using local representations. The trivial solution consists in assigning to test classes the exact same semantic representation as their most similar training class. Consider applying these semantic embeddings within a ZSL framework to our toy problem: classifying a test image x as a Horse relatively to the negative set of TV within the training set becomes strictly equivalent to classifying x as Zebra relatively to its negative set PC within the test set. Hence, any existing ZSL model using these

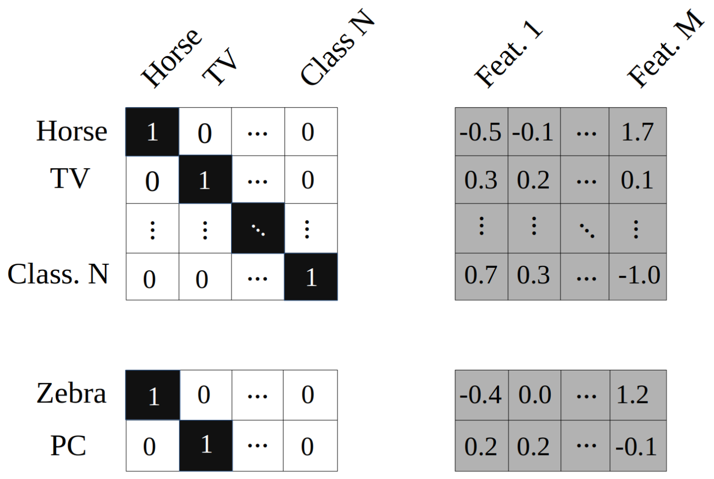

Figure 3.10: Illustration of local (one-hot, on the left) and distributed (right) representations of visual classes. The similarity-based solution encodes both training and test classes as local representations. Composition-based solutions need distributed representations.

local embeddings instead of distributed representations like word embeddings Y would converge to the same solution.

3.4.2 Standard benchmark

Besides our toy example, how well would this trivial solution perform on the standard benchmark?

To apply the trivial solution of the toy example to the standard benchmark, we need a similarity mapping f between training and test classes. To define such mapping, we used the shortest path length between nodes of the Wordnet hierarchy as a measure of distance d. We assign to test classes the semantic embedding of their closest training class, as formalized in equations (4.):

$$f : C_{te} \rightarrow C_{tr} \tag{3.1a}$$

$$f : c \rightarrow argmin_{c' \in C_{tr}} d(c, c') \tag{3.1b}$$

$$y_c = y_{f(c)} + e, \forall c \in C_{te} \tag{3.1c}$$

However, this procedure leads to many test classes sharing the exact same semantic representations. Consider the example of *Cathartid* and *Aegypiidae* classes in Figure 3.1. Both classes are closest to the *Vulture* training classes so they share the same semantic vector $y_{Voluture}$ This leads to undefined behaviors in the classification process. To differentiate between such classes, we add a small Gaussian noise e to the semantic embeddings of test classes, following equation (3.1).

The trivial solution can be implemented by any existing ZSL model using these semantic embeddings. The results reported in Table 3.2 were computed using the Linear baseline model [16]. This table compares the accuracy of this trivial solution to state of the art models as reported in [4, 5]. The trivial similarity-based solution outperforms existing ZSL models by a significant margin. Only GCN-based models [5], which we discuss in the next section, seem to outperform our trivial solution.

Table 3.2: Top-1 accuracy on the standard test splits (top) as reported for linear baselines in [4], (middle) as reported for GCN-based models in [5] and (down) obtained by our trivial solution

model	1-hop	2-hops	all
SYNC [26]	9.26	2.29	0.96
CONSE [27]	7.63	2.18	0.95
ESZSL [16]	6.35	1.51	0.62
LATEM [36]	5.45	1.32	0.5
DEVISE[3]	5.25	1.29	0.49
CMT [64]	2.88	0.67	0.29
GCNZ [65]	19.8	4.1	1.8
ADGPM [5]	**26.6**	**6.3**	**3.0**
Trivial	20.27	3.59	1.53

Comparison of the Trivial solution accuracy to state of the art on the standard benchmark. Top: baseline models as reported in [4]. Middle: s.o.a GCN-based models as reported in [5]

Why does such a simple solution outperform a decade of development effort in ZSL? In the next section, we argue that the standard benchmark has a strong structural bias that favors similarity-based ZSL models.

3.4.3 Measuring structural bias

In our toy example, we have hinted at the fact that structural bias emerges for test sets in which test classes are relatively similar to training classes, while being comparably more dissimilar to each other (to their negative set). To confirm this intuition, we define the following structural ratio:

$$r(c) = \frac{min_{c' \in C_{tr}} d(c, c')}{min_{c' \in C_{te}} d(c, c')} \tag{3.2a}$$

$$R(C_{te}) = \frac{1}{|C_{te}|} \sum_{c \in C_{te}} r(c) \tag{3.2b}$$

In which c represents a visual class, C_{te} and C_{tr} represent test and training sets respectively, and d is a distance reflecting similarity between two classes. Here, $r(c)$ represents the ratio of the distance between c and its closest training class to the distance between c and its closest test class. In our experiments, we use the the shortest path length between two classes in the Wordnet hierarchy as a measure of distance d, although different metrics would be interesting to investigate as well. We compute the structural ratio of a test set $R(C_{te})$ as the mean structural ratio of its individual classes.

Figure 3.11 shows the top-1 accuracy achieved by baseline models on different test sets with respect to their structural ratio R. As for previous experiments, we report our results on test splits of 100 classes.

On test splits of low structural ratio, the trivial solution performs remarkably well, on par with the state of the art GCN model. Such test splits are similar to the toy example in which each test class is closely related to a training class while being far away from other test classes in the Wordnet hierarchy. As an example, the structural ratio of the test split in our toy example is $R(C_{te}) = 1/2 \times (2/4 + 2/4) = 0.5$, which corresponds to the highest accuracies achieved

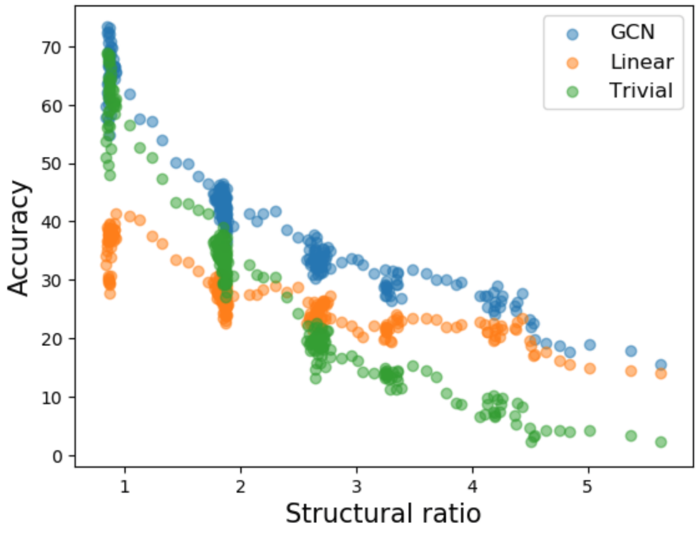

Figure 3.11: ZSL accuracy on different test sets with respect to their structural ratio $R(C_{te})$.

by the trivial solution. We say that such test split is structurally biased towards similarity-based trivial solutions.

However, the accuracy of the similarity-based trivial solution decreases sharply with the structural ratio until it reaches near chance accuracy for the highest ratios. Hence maximizing the structural ratio of test splits seems to be an efficient way to minimize structural bias. Although their accuracy decrease with larger structural ratios, both GCN and Linear models remain well above chance. These results suggest that ZSL models based on word embeddings are indeed capable of compositional reasoning. At the very least, they are able to perform more complex ZSL tasks than the trivial similarity-based solution. Interestingly, as the trivial solution converges towards chance accuracy, the GCN model accuracy seems to converge towards the accuracy of the ZSL baseline. This suggests that the main reason behind the success of GCN models is that they efficiently leverage the Wordnet hierarchy to exploit structural bias.

The *1-hop* and *2-hops* test splits of the standard benchmark consist of the set of test classes closest to the training classes within the Wordnet hierarchy. This leads to test splits of very low structural ratio, similar to our toy example. For instance, the *1-hop* test split has a structural ratio of 0.55. It is an example of structural bias even more extreme than our toy example as test classes are either children or parent classes of a training class. In the next section, we propose a new benchmark with maximal structural ratio in order to minimize structural bias.

3.5 New Benchmark

3.5.1 Proposed Benchmark

In this section, we briefly detail the semi-automated construction of a new benchmark designed to fix the different flaws of the current benchmark highlighted by our analysis. The selection of this new test set proceeds in two steps:

In a first step, we select a subset of candidate test classes $C' \subset C$ from the remaining 20,845 Imagenet classes based on the statistics of image samples and word labels: We first filter out semantic samples $Y' \subset Y$ corresponding to rare or polysemous words of secondary meaning (see Section 2.3). We then discard visual classes of low sample population and filter out ambiguous image samples using supervised learning to select $X' \subset X$. The set of candidate test classes is the subset of visual classes $C' \subset C$ for which sufficiently high quality image and semantic samples were selected.

In a second step, we define the test split $C_{te} \subset C'$ as a *structurally consistent* set of *minimal structural bias*: The test set was carefully selected so as to contain no overlap among its own classes nor with the training classes in order to provide a structurally consistent test set for the generalized ZSL setting. This test set consists of 500 classes of maximal structural ratio $R(C_{te})$ so as to minimize structural bias.

Table 3.3 summarizes the different steps of the creation of our benchmark and details their level of automation, parameters, and the approximate ratio of visual classes selected within each of these steps.

The majority of the visual classes filtered out from our benchmark were automatically discarded based on their weak semantic features, low sample population or structural constraints to avoid both parents and children classes be included

Table 3.3: Summary of the benchmark construction steps

	Step	Automation	Parameters	Filter ratio
Semantic	Frequency	Auto	$f > 500$	82%
	Polysemy	Auto	-	91%
Visual	Class-wise	Auto	$n > 300$	63%
	Sample-wise	Auto	$n_C = 1000$, $n_{tr} = 250$	100%
	Shape	Manual	-	95-99%
	Scale	Manual	-	99%
Structural	Hierarchy	Auto	-	82%
	Mutual Exclusivity	Manual	-	95-99%

in the test set. Only the semantic and visual sample selection steps are parameterized. We select word labels occurring at least 500 times within the Wikipedia corpus to avoid rare words. We only select visual classes with a sample population superior to 300 images.

3.5.2 Evaluation

Table 3.4: Evaluation on the proposed benchmark. Accuracy in the generalized ZSL setting are reported as harmonic means over training and test accuracy following [4]

Model	ZSL		G-ZSL	
	@1	@5	@1	@5
Trivial	1.2	3.9	0	0
CONSE [27]	10.65	25.10	0.12	19.34
DEVISE [3]	11.15	29.52	**7.87**	26.10
ESZSL [16]	13.54	32.61	4.59	25.53
GCN-6 [65]	9.58	27.19	4.81	23.35
GCN-2 [5]	14.09	35.12	4.96	**30.35**
ADGPM [5]	**14.10**	**36.03**	4.90	29.96

Table 3.4 presents the evaluation of a number of baseline models on the newly proposed benchmarks. A few notable results stand out from this table: First, dif-

ferent from the standard benchmark, CONSE [27] performs worse than DEVISE [3]. The relatively high accuracy reported by the CONSE model on the standard benchmark is most likely due to the fact that word embeddings of test classes are statistically close to the word embedding of their parent/children test classes so that CONSE results more closely fit the trivial similarity-based trivial solution. We expect model averaging methods to benefit the most from the structural bias in the standard benchmark.

Second, the impressive improvements reported by GCN-based models over linear baselines are significantly reduced, although GCN models still outperform linear baselines. This result corroborates the observation, in Section 3.4, that GCN models tend to converge towards the results of linear baseline models for high structural ratio.

3.6 Chapter Conclusion

Datasets are the substrat on which algorithmic progress is built. A flawed and biased benchmark will invariably lead to biased algorithmic development. In this section, we have shown major flaws in the standard generic object ZSL benchmark and proposed a new benchmark to address these flaws.

More importantly, we propose that ZSL models should aim to develop the ability to compose visual features to define and recognize new classes. This compisition ability is a requirement to achieve combinatorial generalization across visual classes. We introduced the notion of structural bias in ZSL dataset that allows for zero-shot recognition of visual classes without the need for compositional abilities. We showed that the current standard benchmark presents high structural bias, favoring solutions based on simple similarity matching in semantic space and showed that the current benchmark has favored the development of models maximally exploiting structural bias.

Chapter 4

Visual Feature Extraction

The related publications for this chapter are [76].

4.1 Motivation

In this final chapter, we turn our attention to the visual feature extraction module of the architecture depicted in Figure 1.1. CNNs have become the de-facto architecture for visual feature extraction. However, one major disadvantage of CNN for visual feature extraction is their resource consumption: Training deep models within a reasonable amount of time requires special Graphical Processing Units (GPU) with numerous cores and large memory capacity. In this chapter, we focus on a particular aspect of resource efficiency: optimizing the memory cost of training CNNs.

Beyond its application for ZSL research, we envision several additional benefits from the ability to train large neural networks within limited memory:

Democratization of Deep Learning research: Training large CNN requires special GPUs with large memory capacity. Typical desktop GPUs memory capacity is too small for training large CNNs. As a result, getting into deep learning research comes with the barrier cost of either buying specialized hardware or renting live instances from cloud service providers. Reducing the memory cost of deep model training would allow training deep nets on standard graphic cards without the need for specialized hardware, effectively removing this barrier cost. In this paper, we demonstrate efficient training of a CNN on the CIFAR10 dataset (93.3% accuracy within 67 minutes) on an Nvidia GTX750 with only 1GB of memory.

On-device training: With mobile applications, a lot of attention has been given to optimize inference on edge devices with limited computation resources. Training state-of-the-art CNN on embedded devices, however, has still received little attention. Efficient on-device training is a challenging task for the underlying power efficiency, computation and memory optimization challenges it involves. As such, CNN training has thus far been relegated to large cloud servers, and trained CNNs are typically deployed to embedded device fleets over the network. On-device training would allow bypassing these server-client interactions over the network. We can think of several potential applications of on-device training, including:

- Life-long learning: Autonomous systems deployed in evolving environments like drones, robots or sensor networks might benefit from continuous life-long learning to adapt to their changing environment. On-device training would enable such application without the expensive communication burden of having edge devices continuously sending their data to remote servers over the network. It would also provide resilience to network failures in critical application scenarios.

- In privacy-critical applications such as biometric mobile phone authentication, users might not want to have their data sent over the network. On-device training would allow fine-tuning recognition models on local data without sending sensitive data over the network.

In this work, we propose an architecture with minimal training memory cost requirements which enables training within the tight memory constraints of embedded devices.

Research in optimization: Recent works on stochastic optimization algorithms have highlighted the benefits of large batch training [77, 78]. For example, in Imagenet, linear speed-ups in training have been observed with increasing batch sizes up to tens of thousands of samples [78]. Optimizing the memory cost of CNN training may allow further research on the optimization trade-offs of large batch training. For small datasets like MNIST or CIFAR10, we are able to process the full dataset in 14 and 18 GB of memory respectively. Although large batch training on such small dataset is very computationally inefficient with current stochastic optimization algorithms [78], the ability to process the full dataset in one pass allows to easily train CNNs on the true gradient of the error. Memory optimization techniques have the potential to facilitate research on optimization techniques outside the realm of Stochastic Gradient Descent to be investigated.

In this section, we build on recent works on reversible networks [79, 80] and ask the question: how far can we reduce CNN training memory cost using reversible designs with minimal impact on the accuracy and computational cost? To do so, we analyze the memory cost reduction of different CNN architectures and identify their memory bottleneck, which leads us to introduce a layer-wise invertible architecture. However, we observe that layer-wise invertible networks accumulate numerical errors across their layers, which leads to numerical instabilities impacting model accuracy. We characterize the accumulation of numerical errors within long chains of revertible operations and investigate their effect on model accuracy. To mitigate the impact of these numerical errors on the model accuracy, we propose both a reparameterization of invertible layers and a hybrid architecture combining the benefits of layer-wise and residual-block-wise reversibility to stabilize training. We present a new architecture that allows to efficiently train a CNN with the minimal memory cost of 352 bytes per pixel.

4.2 Related Work

4.2.1 Reversibility

Reversible network designs have been proposed for various purposes including generative modeling, visualization, solving inverse problems, or theoretical analysis of hidden representations.

Flow-based generative models use analytically invertible transformations to compute the change of variable formula. Invertibility is either achieved through channel partitioning schemes (NICE [81] Real-NVP [82]), weight matrix factorization (GLOW [83]) or constraining layer architectures to easily invertible unitary operations (Normalization flows [84])

Neural ODEs [85] take a drastically different take on invertibility: They leverage the analogy between residual networks and the Euler method to define continuous hidden state systems. The conceptual shift from a finite set of discrete transformations to a continuous regime gives them invertibility for free. The computational efficiency of this approach, however, remains to be demonstrated.

The RevNet model [79] was inspired by the Real-NVP generative model. They adapt the idea of channel partitioning and propose an efficient architecture for discriminative learning. The iRevNet [80] model builds on the RevNet architecture: they propose to replace the irreversible max-pooling operation with an invertible operation that reshapes the hidden activation states so as to compensate the loss of spatial resolution by an increase in the channel dimension. By preserving the volume of activations, their pooling operation allows for exact reconstruction of the inverse. In their original work, the authors focus on the analysis of the representations learned by invertible models rather than resource efficiency. From a resource optimization point of view, one downside of their method is that the proposed invertible pooling scheme drastically increases the number of channels in upper layers. As the size of the convolution kernel weights grows quadratically in the number of channels, the memory cost associated with storing the model weights becomes a major memory bottleneck. We address this issue in our proposed architecture. In [86], the authors use these reversible architectures to study undesirable invariances in feature space.

In [87], the authors propose a unified architecture performing well on both generative and discriminative tasks. They enforce invertibility by regularizing the weights of residual blocks so as to guarantee the existence of an inverse operation. However, the computation of the inverse operation is performed with power iteration methods which are not optimal from a computational perspective.

Finally, [88] propose to reconstruct the input activations of normalization and activation layers using their inverse function during the backward pass. We propose a similar method for layer-wise invertible networks. However, as their model does not invert convolution layers, it does not feature long chains of invertible operations so that they do not need to account for numerical instabilities. Instead, our proposed model features long chains of invertible operations so that we need to characterize numerical errors in order to stabilize training.

4.2.2 Resource efficiency

Research into resource optimization of CNNs covers a wide array of techniques, most of which are orthogonal to our work. We briefly present some of these works:

On the architectural side, Squeezenet [89] was first proposed as an efficient neural architecture reducing the number of model parameters while maintaining high classification accuracy. MobileNet [90] uses depth-wise separable convolutions to further reduce the computational cost of inference for embedded device applications.

Network pruning [91] is a set of techniques developed to decrease the model weight size and computational complexity. Network pruning works by removing the network weights that contribute the least to the model output. Pruning deep models has been shown to drastically reduce the memory cost and computational cost of inference without significantly hurting model accuracy. Although pruning has been concerned with optimization of the resource inference, the recently proposed lottery ticket hypothesis [92] has shown that specifically pruned networks could be trained from scratch to high accuracy. This may be an interesting and complementary line of work to investigate in the future to reduce training memory costs.

Low precision arithmetic has been proposed as a mean to reduce both memory consumption and computation time of deep learning models. Mixed precision training [93] combines float16 with float32 operations to avoid numerical instabilities due to either overflow or underflow. For inference, integer quantization [94, 95] has been shown to drastically improve the computation and memory efficiency and has been successfully deployed on both edge devices and data centers. Integrating mixed-precision training to our proposed architecture would allow us to further reduce training memory costs.

Most related to our work, gradient checkpointing was introduced as a mean to reduce the memory cost of deep neural network training. Gradient checkpointing, first introduced in [96], trades off memory for computational complexity by storing only a subset of the activations during the forward pass. During the backward pass, missing activations are recomputed from the stored activations as needed by the backpropagation algorithm. Follow-up work [97] has since built on the original gradient checkpointing algorithm to improve this memory/computation trade-off. However, reversible models like RevNet have been shown to offer better computational complexity than gradient checkpointing, at the cost of constraining the model architecture to invertible residual blocks.

4.3 Preliminaries

In this section, we analyze the memory footprint of training architectures with different reversibility patterns. We start by introducing some notations and briefly review the backpropagation algorithm in order to characterize the training memory consumption of deep neural networks. In our analysis, we use a Resnet-18 as a reference baseline and analyze its training memory footprint. We then gradually augment the baseline architecture with reversible designs and analyze their impact on computation and memory consumption.

4.3.1 Backpropagation & Notations

Let us consider a model F made of N sequential layers trained to minimize the error e defined by a loss function $\mathcal{L}$ for an input x and ground-truth label $\bar{y}$:

$$F : x \rightarrow y \tag{4.1a}$$

$$y = f_N \circ ... \circ f_2 \circ f_1(x) \tag{4.1b}$$

$$e = \mathcal{L}(y, \bar{y}) \tag{4.1c}$$

During the forward pass, each layer f_i takes as input the activations z_{i-1} from the previous layer and outputs activation features $z_i = f_i(z_{i-1})$, with $z_0 = x$ and $z_N = y$ being the input and output of the network respectively.

During the backward pass, the gradient of the loss with respect to the hidden activations are propagated backward through the layers of the networks using the chain rule as:

$$\frac{\delta \mathcal{L}}{\delta z_{i-1}} = \frac{\delta \mathcal{L}}{\delta z_i} \times \frac{\delta z_i}{\delta z_{i-1}} \tag{4.2}$$

Before propagating the loss gradient with respect to its input to the previous layer, each parameterized layer computes the gradient of the loss with respect to its parameters. In vanilla SGD, for a given learning rate η, the weight gradients are subsequently used to update the weight values as:

$$\frac{\delta \mathcal{L}}{\delta \theta_i} = \frac{\delta \mathcal{L}}{\delta z_i} \times \frac{\delta z_i}{\delta \theta_i} \tag{4.3a}$$

$$\theta_i \leftarrow \theta_i - \eta \times \frac{\delta \mathcal{L}}{\delta \theta_i} \tag{4.3b}$$

However, the analytical form of the weight gradients are functions of the layer's input activations z_{i-1}. In convolution layers, for instance, the weight gradients can be computed as the convolution of the input activation by the output's gradient:

$$\frac{\delta \mathcal{L}}{\delta \theta_i} = z_{i-1} \star \frac{\delta \mathcal{L}}{\delta z_i} \tag{4.4}$$

Hence, computing the derivative of the loss with respect to each layer's parameters θ_i requires knowledge of the input activation values z_{i-1}. In the standard backpropagation algorithm, hidden layers activations are stored in memory upon computation during the forward pass. Activations accumulate in live memory buffers until used for the weight gradients computation in the backward pass. Once the weight gradients computed in the backward pass, the hidden activation buffers can be freed from live memory. However, the accumulation of activation values stored within each parameterized layer along the forward pass creates a major bottleneck in GPU memory.

The idea behind reversible designs is to constrain the network architecture to feature invertible transformations. Doing so, activations z_i in lower layers can be recomputed through inverse operations from the activations $z_{j>i}$ of higher layers. In such architectures, activation do not need to be kept in memory during the forward pass as they can be recomputed from higher layer activations during the backward pass, effectively freeing up the GPU live memory.

4.3.2 Memory footprint

We denote the memory footprint of training a neural network as a value $\mathcal{M}$ in bytes. Given an input x and ground truth label $\bar{y}$, the memory footprint represents the peak memory consumption during an iteration of training including the forward and backward pass. We divide the total training memory footprint $\mathcal{M}$ into several memory cost factors: the cost M_θ of storing the model weights, the hidden activations M_z, and the gradients M_g:

$$\mathcal{M} = M_\theta + M_z + M_g \tag{4.5}$$

In the following subsections, we detail the memory footprint of existing architectures with different reversibility patterns. To help us formalize these memory costs, we further introduce the following notations: let $n(x)$ denote the number of elements in a tensor x, i.e.; if x is an $h \times w$ matrix, then $n(x) = h \times w$. Let *bpe* be the memory cost in bytes per elements of a given precision so that the actual memory cost for storing an $h \times w$ matrix is $n(x) \times bpe$. For instance, float32 tensors have a memory cost per element $bpe = 4$. We use *bs* to denote the batch size, and c_i to denote the number of channels at layer i.

4.3.3 Vanilla ResNet

The architecture of a vanilla ResNet-18 is shown in Figure 4.1. Vanilla ResNets do not use reversible computations so that the input activations of all parameterized layers need to be accumulated in memory during the forward pass for the computation of the weight gradients to be done in the backward pass.

Hence the peak memory footprint of training a vanilla ResNet happens at the beginning of the backward pass when the top layer's activation gradients need to be stored in memory in addition to the full stack of hidden activation values.

Let us denote by $P \subset N$ the subset of parameterized layers of a network F (i.e.; convolutions and batch normalization layers, excluding activation functions and

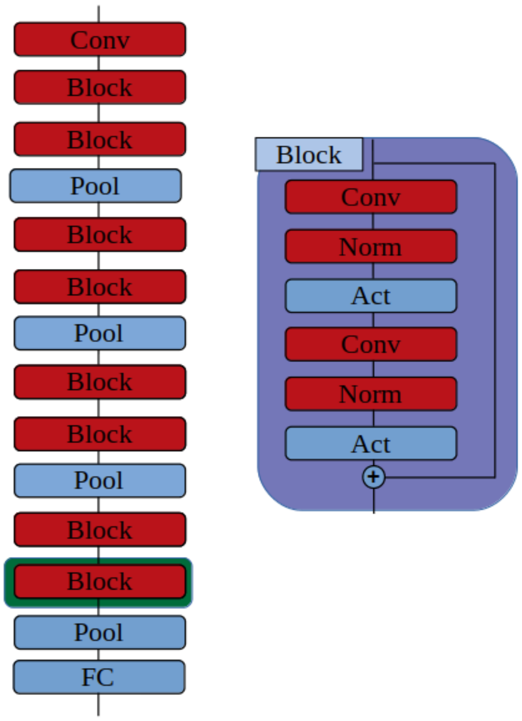

Figure 4.1: Illustration of the ResNet-18 architecture and its memory requirements. Modules contributing to the peak memory consumption are shown in red. These modules contribute to the memory cost by storing their input in memory. The green annotation represents the extra memory cost of storing the gradient in memory. The peak memory consumption happens in the backward pass through the last convolution so that this layer is annotated with an additional gradient memory cost. At this step of the computation, all lower parameterized layers have stored their input in memory, which constitutes the memory bottleneck.

pooling layers). The memory cost associated with storing the hidden activation values is given by:

$$M_z = \sum_{i \in P} n(z_i) \times bpe \tag{4.6a}$$

$$= \sum_{i \in P} bs \times c_i \times h_i \times w_i \times bpe \tag{4.6b}$$

Where h_i and w_i represent the spatial dimensions of the activation values at layer i. h_i and w_i are determined by the input image size $h \times w$ and the pooling factor p_i of layer i, so we can factor out both the spatial dimensions and the batch size from this equation, yielding a memory cost per input pixel M'_z:

$$\begin{aligned} M_z &= \sum_{i \in P} bs \times h \times w \times p_i \times c_i \times bpe && (4.7a) \\ &= bs \times h \times w \times \sum_{i \in P} p_i \times c_i \times bpe && (4.7b) \\ M'_z &= \frac{M_z}{bs \times h \times w} && (4.7c) \\ &= \sum_{i \in P} p_i \times c_i \times bpe && (4.7d) \end{aligned}$$

The memory footprint of the weights is given by:

$$M_\theta = \sum_{i \in P} n(\theta_i) \times bpe \tag{4.8}$$

The memory footprint of the gradients correspond to the size of the gradient buffers at the time of peak memory usage. In a vanilla ResNet18 model, this peak memory usage happens during the backward pass through the last convolution of the network. Hence, the memory footprint of the gradients correspond to the memory cost of storing the gradients with respect to either the input or the output of this layer, which also depends on the input pixel size:

$$\begin{aligned} M_g &= max(n(g_{i-1}), n(g_i)) \times bpe && (4.9a) \\ &= h \times w \times bs \times p_i \times max(c_{i-1}, c_i) \times bpe && (4.9b) \\ M'_g &= p_i \times max(c_{i-1}, c_i) \times bpe && (4.9c) \end{aligned}$$

Figure 4.1 illustrates the peak memory consumption of a ResNet-like architecture. For a ResNet parameterized following Table 1, the peak memory consumption can then be computed as:

$$\begin{aligned}\mathcal{M} &= M_\theta + M_z + M_g && \text{(4.10a)}\\ &= M_\theta + (M'_z + M'_g) \times (h \times w \times bs) && \text{(4.10b)}\\ &= 12.5 * 10^6 + 1928 \times (h \times w \times bs) && \text{(4.10c)}\\ & && \text{(4.10d)}\end{aligned}$$

For example, a training iteration over a typical batch of 32 images of resolution 240×240 requires 12.5 MB of memory to store the model weights and 3.8 GB of memory to store the hidden layers activations and gradients for a total of $\mathcal{M} = 3.81$ GB of VRAM. The memory cost of the hidden activations is thus the main memory bottleneck of CNN training as the cost associated with the model weights is negligible in comparison.

4.3.4 RevNet

The RevNet architecture introduces reversible blocks as drop-in replacements of the residual blocks of the ResNet architecture. Reversible blocks have analytical inverses that allow for the computation of both their input and hidden activation values from the value of their output activations. Two factors create memory bottlenecks in training RevNet architectures, which we refer to as the local and global bottlenecks.

First, the RevNet architecture features non-volume preserving max-pooling layers, for which the inverse cannot be computed. As these layers do not have analytical inverses, their input must be stored in memory during the forward pass for the reconstruction of lower layer's activations to be computed during the backward pass. We refer to the memory cost associated with storing these activations as the global bottleneck, since these activations need to be accumulated during the forward pass through the full architecture.

The local memory bottleneck has to do with the synchronization of the reversible block computations: While activations values are computed by a forward pass through the reversible block modules, gradients computations flow backward through these modules so that the activations and gradient computations cannot be performed simultaneously. Figure 4.2 illustrates the process of backpropagating through a reversible block: First, the input activation values of the parameterized hidden layers within the reversible blocks are recomputed from the output. Once the full set of activation have been computed and stored in GPU memory, the backpropagation of the gradients through the reversible block can begin. We refer to the accumulation of the hidden activation values within the reversible block as the local memory bottleneck.

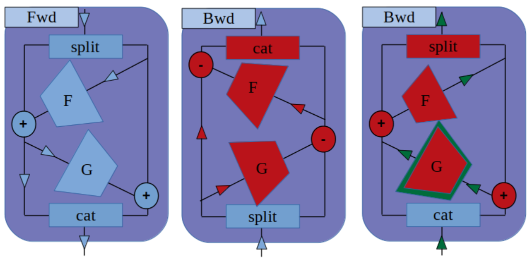

Figure 4.2: Illustration of the backpropagation process through a reversible block. In the forward pass (left), activations are propagated forward from top to bottom. The activations are not kept in live memory as they are to be recomputed in the backward pass so no memory bottleneck occurs. The backward pass is made of two phases: First the hidden and input activations are recomputed from the output through an additional forward pass through both modules (middle). Once the activations recomputed, the activations gradient are propagated backward through both modules of the reversible blocks (right). Because the activation and gradient computations flow in opposite directions through both modules, both computations cannot be efficiently overlapped, which results in the local memory bottleneck of storing all hidden activations within the reversible block before the gradient backpropagation step.

For a typical parameterization of a RevNet, as summarized in Table 1, the local bottleneck of lower layers actually outweighs the global memory bottleneck introduced by non-reversible pooling layers. Indeed, as the spatial resolution decreases with pooling operations, the cost associated with storing the input activations of higher layers becomes negligible compared to the cost of storing activation values in lower layers.

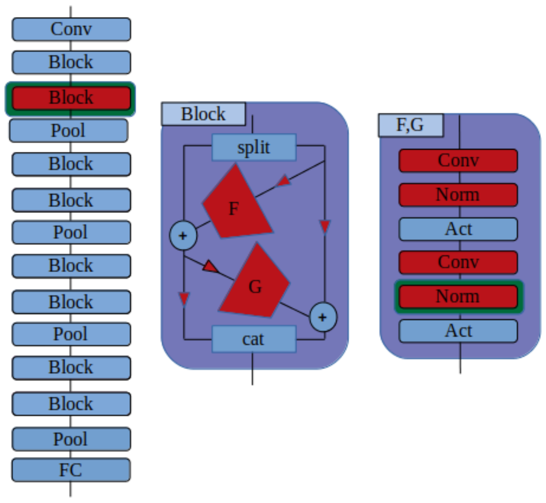

Figure 4.3: Illustration of the Revnet architecture and its memory consumption. Modules contributing to the peak memory consumption are shown in red. The peak memory consumption happens during the backward pass through the first reversible block. At this step of the computations, all hidden activations within the reversible block are stored in memory simultaneously.

Hence, surprisingly, the peak memory consumption of the RevNet architecture, as illustrated in Figure 4.3, happens in the backward pass through the first reversible block, in which the local memory bottleneck is maximum. For the architecture described in Table 1, the peak memory consumption can be computed as:

$$\mathcal{M} = M_\theta + M_z + M_g \tag{4.11a}$$
$$= (M_\theta + (M'_z + M'_g) \times (h \times w \times bs) \tag{4.11b}$$
$$= 12.7 \times 10^6 + 640 \times (h \times w \times bs) \tag{4.11c}$$

Following our previous example, a RevNet architecture closely mimicking the ResNet-18 architecture requires $\mathcal{M} = 1.19$ GB of VRAM for a training iteration over batch of 32 images of resolution 240×240.

Finally, the memory savings allowed by the reversible block come with the additional computational cost of computing the hidden activations during the backward pass. As noted in the original paper, this computational cost is equivalent to performing one additional forward pass.

4.3.5 iRevNet

The iRevNet model builds on the RevNet architecture: they replace the irreversible max-pooling operation with an invertible operation that reshapes the hidden activation states so as to compensate for the loss of spatial resolution by an increase in the channel dimension. As such, the iRevNet architecture is fully invertible, which alleviates the global memory bottleneck of the RevNet architecture.

This pooling operation works by stacking the neighboring elements of the pooling regions along the channel dimension, i.e.; for a 2D pooling operation with 2×2 pooling window, the number of output channels is four times the number of input channels. Unfortunately, the size of a volume-preserving convolution kernel grows quadratically in the number of input channels:

$$M(\theta) = c_{in} \times c_{out} \times k_h \times k_w \tag{4.12a}$$

$$= c^2 \times k_h \times k_w \tag{4.12b}$$

Consider an iRevNet network with initial channel size 32. After three levels of 2×2 pooling, the effective channel size becomes $32 \times 4^3 = 2048$. A typical 3×3 convolution layer kernel for higher layers of such network would have $n(\theta) = 2048^2 \times 3 \times 3 = 37M$ parameters. At this point, the memory cost of the network weights M_θ becomes an additional memory bottleneck.

Furthermore, the iRevNet architecture does not address the local memory bottleneck of the reversible blocks. Figure 4.4 illustrates such architecture. For an initial channel size of 32, as summarized in Table 1, the peak memory consumption is given by:

$$\mathcal{M} = M_\theta + M_z + M_g \tag{4.13a}$$

$$= M_\theta + (M'_z + M'_g) \times (h \times w \times bs) \tag{4.13b}$$

$$= 171 \times 10^6 + 640 \times (h \times w \times bs) \tag{4.13c}$$

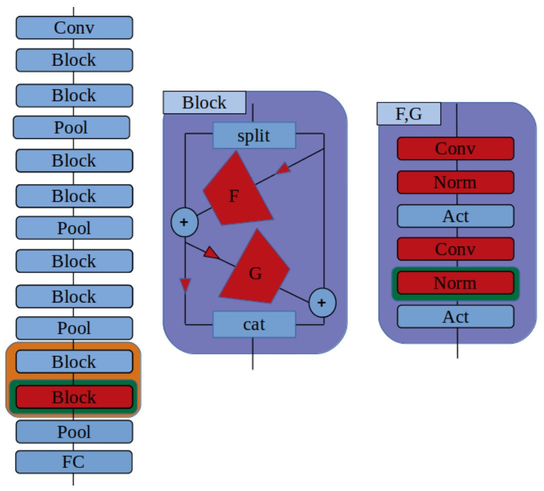

Figure 4.4: Illustration of the i-Revnet architecture and its memory consumption. The peak memory consumption happens during the backward pass through the top reversible block. In addition to this local memory bottleneck, the cost of storing the top layers weights (in orange) becomes a new memory bottleneck as the weight kernel size grows quadratically in the number of channels.

Training such an architecture for an iteration over batches of 32 images of resolution 240×240 would require $\mathcal{M} = 1.35$GB of VRAM. In the next section, we introduce both layer-wise reversibility and a variant on this pooling operations to address the local memory bottleneck of reversible blocks and the weight memory bottleneck respectively.

4.4 Method

RevNet and iRevNet architectures implement reversible transformations at the level of residual blocks. As we have seen in the previous section, the design of these reversible blocks create a local memory bottleneck as all hidden activations within a reversible block need to be computed before the gradients are back-propagated through the block. In order to circumvent this local bottleneck, we introduce layer-wise invertible operations. However, these invertible operations introduce numerical error, which we characterize in the following subsections. In Section 5, we will show that these numerical errors lead to instabilities that degrade the model accuracy. Hence, in section 4.2, we propose a hybrid model combining layer-wise and residual block-wise reversible operations to stabilize training while resolving the local memory bottleneck at the cost of a small additional computational cost.

4.4.1 Layer-wise Invertibility

In this section, we present invertible layers that act as drop-in replacement for convolution, batch normalization, pooling and non-linearity layers. We then characterize the numerical instabilities arising from the invertible batch normalization and non-linearities.

4.4.1.1 Invertible batch normalization

As batch normalization is not a bijective operation, it does admit an analytical inverse. However, the inverse reconstruction of a batch normalization layer can be realized with minimal memory cost. Given first and second order moment parameters β and γ, the forward f and inverse f^{-1} operation of an invertible batch normalization layer can be computed as follows:

$$y = f(x) = \gamma \times \frac{x - \hat{x}}{\sqrt{\dot{x}} + \epsilon} + \beta \tag{4.14a}$$

$$x = f^{-1}(y, \hat{x}, \dot{x}) = (\sqrt{\dot{x}} + \epsilon) \times \frac{y - \beta}{\gamma} + \hat{x} \tag{4.14b}$$

Where $\hat{x}$ and $\dot{x}$ represent the mean and variance of x respectively. Hence, the input activation x can be recovered from y through f^{-1} at the minimal memory cost of storing the input activation statistics $\hat{x}$ and $\dot{x}$.

Let us consider the accumulation of numerical errors arising from the inverse computation of an invertible batch normalization layer. During the backward pass, the invertible batch norm layer is supposed to compute its input $x = f^{-1}(y, \hat{x}, \dot{x})$ from the output y. In reality, however, the output recovered by upstream invertible layers is a noisy estimate $\hat{y} = y + \epsilon_y$ of the true output due to numerical errors introduced by upstream layers. Let us define the signal to noise ratio (SNR) of the input and output signal as follows:

$$snr_o = \frac{|y|^2}{|\epsilon^y|^2} \tag{4.15a}$$

$$snr_i = \frac{|x|^2}{|\epsilon^x|^2} \tag{4.15b}$$

We are interested in characterizing the factor α of reduction of the SNR through the inverse reconstruction:

$$\alpha = \frac{snr_i}{snr_o} \tag{4.16}$$

To illustrate the mechanism through which the batch normalization inverse operation reduces the SNR, let us consider a toy layer with only two channels and parameters $\beta = [0, 0]$ and $\gamma = [1, \rho]$. For simplicity, let us consider an input signal x independently and identically distributed across both channels with zero mean and standard deviation 1 so that, in the forward pass, we have:

$$y = [y_0, y_1] \tag{4.17a}$$

$$= [x_0, x_1 \times \rho] \tag{4.17b}$$

$$|y|^2 = |x_0|^2 + |x_1|^2 \times \rho^2 \tag{4.17c}$$

$$= \frac{1}{2} \times |x|^2 + \frac{1}{2} \times |x|^2 \times \rho^2 \tag{4.17d}$$

$$= \frac{|x|^2}{2} \times (1 + \rho^2) \tag{4.17e}$$

In which we used the assumption that x is independently and identically distributed across both channels to factorize $|x_0|^2 = |x_1|^2 = \frac{1}{2} \times |x|^2$ in equation (17d).

During the backward pass, the noisy estimate $\tilde{y} = y + \epsilon^y$ is fed back as input to the inverse operation. Similarly, let us suppose a noise ϵ^y identically distributed across both channels so that we have:

$$\begin{aligned}
\tilde{y} =& [\tilde{y}_0, \tilde{y}_1] && (4.18a)\\
=& [x_0 + \epsilon^y_0, x_1 \times \rho + \epsilon^y_1] && (4.18b)\\
\tilde{x} =& [\tilde{y}_0, \frac{\tilde{y}_1}{\rho}] && (4.18c)\\
=& [x_0 + \epsilon^y_0, x_1 + \frac{\epsilon^y_1}{\rho}] && (4.18d)\\
\epsilon^x =& \tilde{x} - x && (4.18e)\\
=& [\epsilon^y_0, \frac{\epsilon^y_1}{\rho}] && (4.18f)\\
|\epsilon^x|^2 =& |\epsilon^y_0|^2 + \frac{|\epsilon^y_1|^2}{\rho^2} && (4.18g)\\
=& \frac{1}{2} \times |\epsilon^y|^2 + \frac{1}{2} \times \frac{|\epsilon^y|^2}{\rho^2} && (4.18h)\\
=& \frac{|\epsilon^y|^2}{2} \times (1 + \frac{1}{\rho^2}) && (4.18i)
\end{aligned}$$

Using the above formulation, the SNR reduction factor α can be expressed as:

$$\begin{aligned}
\alpha =& \frac{snr_i}{snr_o} && (4.19a)\\
=& \frac{|x|^2}{|\epsilon^x|^2} \times \frac{|\epsilon^y|^2}{|y|^2} && (4.19b)\\
=& \frac{4}{(1 + \frac{1}{\rho^2}) \times (1 + \rho^2)} && (4.19c)
\end{aligned}$$

Figure 4.5 shows the expected evolution of α through our toy layer for different values of the factor ρ. To validate our formula, we empirically evaluate α for normal Gaussian inputs x and output noise ϵ^y and find it to closely match the theoretical results given by equation 19.

In essence, numerical instabilities in the inverse computation of the batch normalization layer arise from the fact that the signal across different channels i

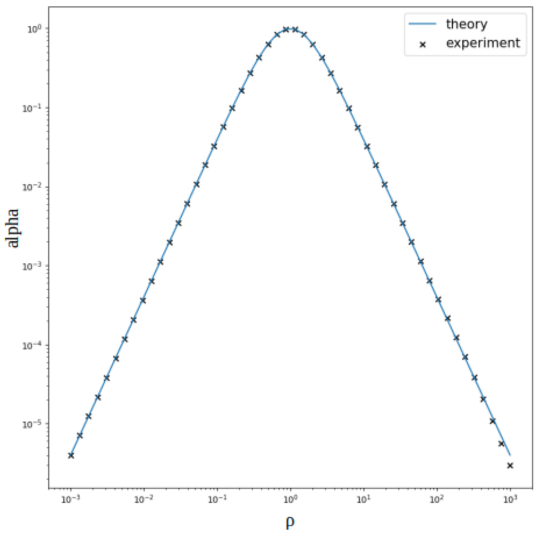

Figure 4.5: Illustration of the numerical errors arising from batch normalization layers. Comparison of the theoretical and empirical evolution of the α ratio for different ρ values in our toy example. Empirical values were computed for a Gaussian input signal with zero mean and standard deviation 1 and a white Gaussian noise of standard deviation 10^{-5}.

and j are amplified by different factors γ_i and γ_j. While the signal amplification in the forward and inverse path cancel out each other ($x = f^{-1}(f(x))$), the noise only gets amplified in the backward pass.

In the above demonstration, we have used a toy parameterization of the invertible batch normalization layer to illustrate the mechanism behind the SNR degradation. For arbitrarily parameterized batch normalization layers, the SNR degradation factor becomes:

$$\begin{aligned}
\alpha =& \frac{snr_i}{snr_o} && (4.20a)\\
=& \frac{|x|^2}{|\epsilon^x|^2} \times \frac{|\epsilon^y|^2}{|y|^2} && (4.20b)\\
=& \frac{|x|^2}{|y|^2} \times \frac{|\epsilon^y|^2}{|\epsilon^x|^2} && (4.20c)
\end{aligned}$$

Assuming a noise ϵ^y, equally distributed across all channels, the noise ratio can be computed as follows:

$$\tilde{y}_i = \gamma_i \times \frac{x_i - \hat{x}_i}{\sqrt{\dot{x}_i} + \epsilon} + \beta_i + \epsilon_i^y \tag{4.21a}$$

$$\tilde{x}_i = (\sqrt{\dot{x}_i} + \epsilon) \times \frac{\tilde{y}_i - \beta_i}{\gamma_i} + \hat{x}_i \tag{4.21b}$$

$$= x_i + \frac{\sqrt{\dot{x}_i} + \epsilon}{\gamma_i} \times \epsilon_i^y \tag{4.21c}$$

$$\epsilon_i^x = \tilde{x}_i - x_i \tag{4.21d}$$

$$= \frac{\sqrt{\dot{x}_i} + \epsilon}{\gamma_i} \times \epsilon_i^y \tag{4.21e}$$

$$\frac{|\epsilon^y|^2}{|\epsilon^x|^2} = \frac{|\epsilon^y|^2}{\frac{|\epsilon^y|^2}{c} \times \sum_i \frac{\dot{x_i}^2}{\gamma_i^2}} \tag{4.21f}$$

$$= \frac{c}{\sum_i \frac{\sqrt{\dot{x}_i} + \epsilon}{\gamma_i}} \tag{4.21g}$$

Assuming input x following a Gaussian distribution with channel-wise mean $\hat{x_i}$ and variance $\dot{x_i}$, the SNR reduction factor α becomes:

$$\frac{|x|^2}{|y|^2} = \frac{\sum_i |x_i|^2}{\sum_i |y_i|^2} \tag{4.22a}$$

$$= \frac{\sum_i (\hat{x}_i^2 + \dot{x}_i)}{\sum_i (\gamma_i^2 + \beta_i^2)} \tag{4.22b}$$

$$\alpha = \frac{|x|^2}{|y|^2} \times \frac{|\epsilon^y|^2}{|\epsilon^x|^2} \tag{4.22c}$$

$$= \frac{\sum_i (\hat{x}_i^2 + \dot{x}_i)}{\sum_i (\gamma_i^2 + \beta_i^2)} \times \frac{c}{\sum_i \frac{\sqrt{\dot{x}_i} + \epsilon}{\gamma_i}} \tag{4.22d}$$

Finally, we propose the following modification, introducing the hyperparameter ϵ_i, to the invertible batch normalization layer:

$$y = f(x) = |\gamma + \epsilon_i| \times \frac{x - \hat{x}}{\sqrt{\dot{x}} + \epsilon} + \beta \tag{4.23a}$$

$$x = f^{-1}(y) = (\sqrt{\dot{x}} + \epsilon) \times \frac{y - \beta}{|\gamma + \epsilon_i|} + \hat{x} \tag{4.23b}$$

The introduction of the ϵ_i hyper parameter serves two purposes: First, it stabilizes the numerical errors described above by lower bounding the smallest γ parameters. Second, it prevents numerical instabilities that would otherwise arise from the inverse computation as γ parameters tend towards zero.

4.4.1.2 Invertible activation function

A good invertible activation function must be bijective (to guarantee the existence of an inverse function) and non-saturating (for numerical stability). For these properties, we focus our attention on Leaky ReLUs whose forward f and inverse f^{-1} computations are defined, for a negative slope parameter n, as follow:

$$y = f(x) = \begin{cases} x, & \text{if } x > 0 \\ x/n, & \text{otherwise} \end{cases} \tag{4.24a}$$

$$x = f^{-1}(y) = \begin{cases} y, & \text{if } y > 0 \\ y \times n, & \text{otherwise} \end{cases} \tag{4.24b}$$

The analysis of the numerical errors yielded by the invertible Leaky ReLU follows a similar reasoning as the toy batch normalization example with an additional subtlety: Similar to the toy batch normalization example, we can think of the leaky ReLU as artificially splitting the input x across two different channels, one channel leaving the output unchanged and one channel that divides the input by a factor n during the forward pass and multiplies its output by a factor n during the backward pass.

However, these artificial channels are defined by the sign of the input and output during the forward and backward pass respectively. Hence, we need to consider the cases in which the noise flips the sign of the output activations, which leads to different behaviors of the invertible Leaky ReLU across four cases:

$$y = \begin{cases} y_{nn} \text{ if } \hat{y} < 0 & \text{and } y < 0 \\ y_{np} \text{ if } \hat{y} >= 0 & \text{and } y < 0 \\ y_{pp} \text{ if } \hat{y} >= 0 & \text{and } y >= 0 \\ y_{pn} \text{ if } \hat{y} < 0 & \text{and } y >= 1 \end{cases} \tag{4.25a}$$

Where the index np, for instance, represents negative activations whose reconstructions have become positive due to the added noise. The signal to noise ratio of the input and outputs can be expressed respectively as:

In the case where $y >> \epsilon_y$, the probability of sign flips (y_{np}, y_{pn}) is negligible, so that the output signal y is evenly split along y_{pp} and y_{nn}. In this regime, the degradation of the SNR obeys a formula similar to the toy batch normalization example:

$$\begin{aligned} y =& [y_{pp}, y_{nn}] && \text{(4.26a)} \\ =& [x_{pp}, \frac{x_{nn}}{n}] && \text{(4.26b)} \\ |y|^2 =& \frac{1}{2} \times |x|^2 + \frac{1}{2} \times \frac{|x|^2}{n^2} && \text{(4.26c)} \\ =& \frac{|x|^2}{2} \times (1 + \frac{1}{n^2}) && \text{(4.26d)} \end{aligned}$$

$$\begin{aligned} \tilde{y} =& [\tilde{y}_{pp}, \tilde{y}_{nn}] && \text{(4.27a)} \\ =& [x_{pp} + \epsilon^y_{pp}, \frac{x_{nn}}{n} + \epsilon^y_{nn}] && \text{(4.27b)} \\ \tilde{x} =& [\tilde{y}_{pp}, \tilde{y}_{nn} \times n] && \text{(4.27c)} \\ =& [x_{pp} + \epsilon^y_{pp}, x_{nn} + \epsilon^y_{nn} \times n] && \text{(4.27d)} \\ \epsilon^x =& \tilde{x} - x && \text{(4.27e)} \\ =& [\epsilon^y_{pp}, \epsilon^y_{nn} \times n] && \text{(4.27f)} \\ |\epsilon^x|^2 =& \frac{1}{2} \times |\epsilon^y|^2 + \frac{1}{2} \times |\epsilon^y|^2 \times n^2 && \text{(4.27g)} \\ =& \frac{|\epsilon^y|^2}{2} \times (1 + n^2) && \text{(4.27h)} \end{aligned}$$

Using the above formulation, the signal to noise ration reduction factor α can be expressed as:

$$\begin{aligned} \alpha =& \frac{snr_i}{snr_o} && (4.28a)\\ =& \frac{|x|^2}{|\epsilon^x|^2} \times \frac{|\epsilon^y|^2}{|y|^2} && (4.28b)\\ =& \frac{4}{(1+\frac{1}{n^2}) \times (1+n^2)} && (4.28c) \end{aligned}$$

Hence numerical errors can be controlled by setting the value of the negative slope n. As n tends towards 1, α converges to 1, yielding minimum signal degradation. However, as n tends towards 1, the network tends toward a linear behavior, which hurts the model expressivity. Figure 4.6 shows the evolution of the SNR degradation α for different negative slopes n; and, in Section 5.1, we investigate the impact of the negative slope parameter on the model accuracy.

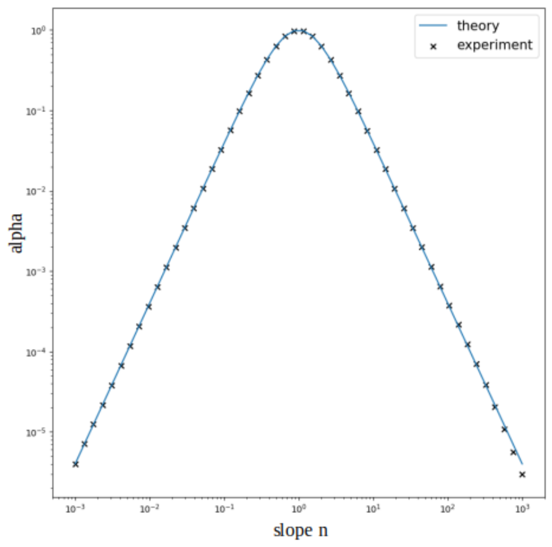

Figure 4.6: Illustration of the numerical errors arising from invertible activation layers. Comparison of the theoretical and empirical evolution of the α ratio for different negative slopes n. Empirical values were computed for a Gaussian input signal with zero mean and standard deviation 1 and a white Gaussian noise of standard deviation 10^{-5}.

When the noise reaches an amplitude similar to or greater than the activation signal, the effects of sign flips complicate the equation. However, in this regime, the signal to noise ratio becomes too low for training to converge, as numerical errors prevent any useful weight update, so we leave the problem of characterizing this regime open.

4.4.1.3 Invertible convolutions

Invertible convolution layers can be defined in several ways. The inverse operation of a convolution is often referred to as deconvolution, and is defined for a subspace of the kernel weight space.

However, deconvolutions are computationally expensive and subject to numerical errors. Instead, we choose to implement invertible convolutions using the channel partitioning scheme as the reversible block design for its simplicity, numerical stability and computational efficiency. Hence, invertible convolutions, in our architecture, can be seen as minimal reversible blocks in which both modules consist of a single convolution. Gomez *et al.*[79] found the numerical errors introduced by reversible blocks to have no impact on the model accuracy. Similarly, we found reversible blocks extremely stable yielding negligible numerical errors compared to the invertible batch normalization and Leaky ReLU layers.

4.4.1.4 Pooling

In [80], the authors propose an invertible pooling operation that operates by stacking the neighboring elements of the pooling regions along the channel dimension. As noted in Section 3.5, the increase in channel size at each pooling level induces a quadratic increase in the number of parameters of upstream convolution, which creates a new memory bottleneck.

To circumvent this quadratic increase in the memory cost of the weight, we propose a new pooling layer that stacks the elements of neighboring pooling regions along the batch size instead of the channel size. We refer to both kind of pooling as channel pooling $\mathcal{P}_c$ and batch pooling $\mathcal{P}_b$ respectively, depending on the dimension along which activation features are stacked. Given a 2×2 pooling region and an input activation tensor x of dimensions $bs \times c \times h \times w$, where bs refers to the batch size, c to the number of channels and $h \times w$ to the spatial resolution, the reshaping operation performed by both pooling layers can be formalized as follows:

$$\begin{aligned} \mathcal{P}_c : & x \to y && \text{(4.29a)} \\ : & \mathbb{R}^{bs \times c \times h \times w} \to \mathbb{R}^{bs \times 4c \times \frac{h}{2} \times \frac{w}{2}} && \text{(4.29b)} \\ \mathcal{P}_b : & x \to y && \text{(4.29c)} \\ : & \mathbb{R}^{bs \times c \times h \times w} \to \mathbb{R}^{4bs \times c \times \frac{h}{2} \times \frac{w}{2}} && \text{(4.29d)} \end{aligned}$$

Channel pooling gives us a way to perform volume-preserving pooling operations while increasing the number of channels at a given layer of the architecture, while batch pooling gives us a way to perform volume-preserving pooling operations while keeping the number of channel constant, By alternating between channel and batch pooling, we can control the number of channels at each pooling level of the model's architecture.

4.4.1.5 Layer-wise invertible architecture

Putting together the above building blocks, Figure 4.7 illustrates a layer-wise invertible architecture. The peak memory usage for a training iteration of this architecture, as parameterized in Table 1, can be computed as follows:

$$\begin{aligned} \mathcal{M} &= M_\theta + M_z + M_g && \text{(4.30a)} \\ &= M_\theta + (M'_z + M'_g) \times (h \times w \times bs) && \text{(4.30b)} \\ &= 29.6 \times 10^6 + 320 \times (h \times w \times bs) && \text{(4.30c)} \end{aligned}$$

Training an iteration over a typical batch of 32 images with resolution 240×240 would require $\mathcal{M} = 590$MB of VRAM. Similar to the RevNet architecture, the reconstruction of the hidden activations by inverse transformations during the backward pass comes with an additional computational cost similar to a forward pass.

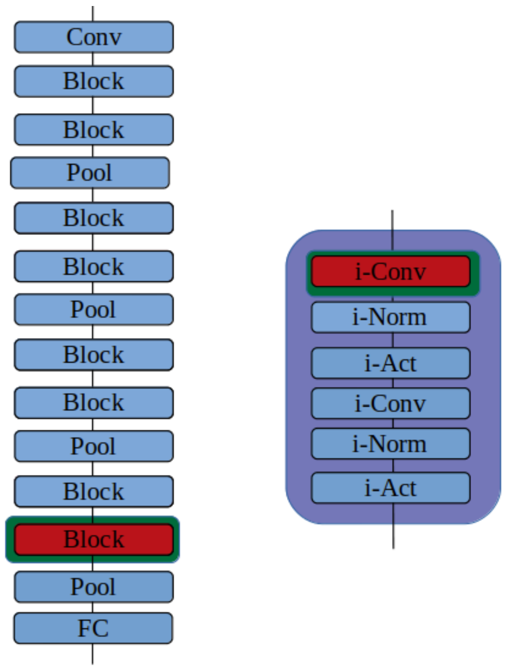

Figure 4.7: Illustration of a layer-wise invertible architecture and its memory consumption.

4.4.2 Hybrid architecture

In section 3, we saw that layer-wise activation and normalization layers degrade the signal to noise ratio of the reconstructed activations. In section 5.1, we will quantify the accumulation of numerical errors through long chains of layer-wise invertible operations and show that numerical errors negatively impact model accuracy.

To prevent these numerical instabilities, we introduce a hybrid architecture, illustrated in Figure 4.8, combining reversible residual blocks with layer-wise invertible functions. Conceptually, the role of the residual level reversible block is to reconstruct the input activation of residual blocks with minimal errors, while the role of the layer-wise invertible layers is to efficiently recompute the hidden activations within the reversible residual blocks at the same time as the gradient propagates to circumvent the local memory bottleneck of the reversible module.

The backward pass through these hybrid reversible blocks is illustrated in Figure 4.9 and proceeds as follows: First, the input x is computed from the output

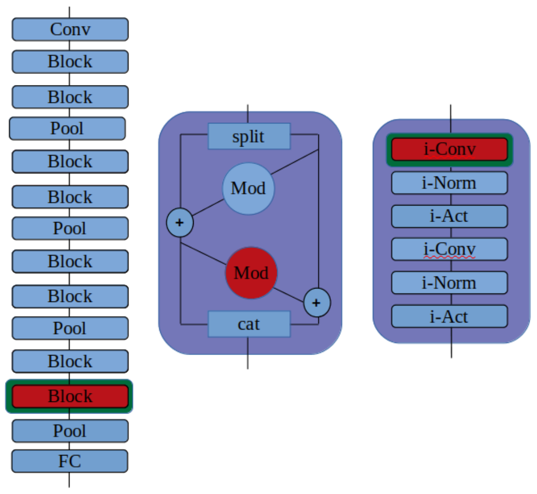

Figure 4.8: Illustration of a hybrid architecture and its peak memory consumption.

y through the analytical inverse of the reversible block. These computations are made without storing the hidden activation values of the sub-modules. Second, the gradient of the activations are propagated backward through the reversible of the block modules. As each layer within these modules is invertible, the hidden activation values are computed using the layer-wise inverse along the gradient.

The analytical inverse of the residual level reversible blocks is used to propagate hidden activations with minimal reconstruction error to the lower modules, while layer-wise inversion allows us to alleviate the local bottleneck of the reversible block by computing the hidden activation values together with the backward flow of the gradients. As layer-wise inverses are only used for hidden feature computations within the scope of the reversible block, and reversible blocks are made of relatively short chains of operations, numerical errors do not accumulate up to a damaging degree.

The peak memory consumption of our proposed architecture, as illustrated in Figure 4.8 and parameterized in Table 1, can be computed as

$$\begin{aligned} \mathcal{M} &= M_\theta + M_z + M_g && (4.31a)\\ &= M_\theta + (M'_z + M'_g) \times (h \times w \times bs) && (4.31b)\\ &= 14.8 \times 10^6 + 352 \times (h \times w \times bs) && (4.31c) \end{aligned}$$

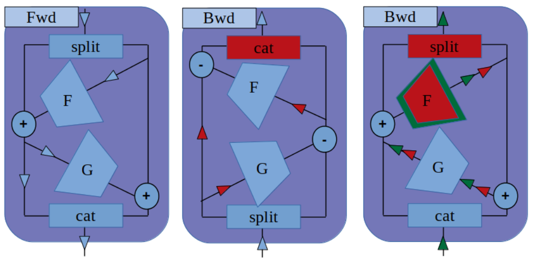

Figure 4.9: Illustration of the backpropagation process through a reversible block of our proposed hybrid architecture. In the forward pass (left), activations are propagated forward from top to bottom. The activations are not kept in live memory as they are to be recomputed in the backward pass so that no memory bottleneck occurs. The backward pass is made of two phases: First the input activations are recomputed from the output using the Reversible block analytical inverse (middle). This step allows to reconstruct the input activations with minimal reconstruction error. During this step, hidden activations are not kept in live memory so as to avoid the local memory bottleneck of the reversible block. Once the input activation recomputed, the gradients are propagated backward through both modules of the reversible blocks (right). During this second phase, hidden activations are recomputed backward through each module using the layer-wise inverse operations, yielding minimal memory footprint

Training an iteration over batch of 32 images of resolution 240×240 would require $\mathcal{M} = 648$MB of VRAM.

It should be noted, however, that this architecture adds an extra computational cost as both the reversible block inverse and layer-wise inverse need to be computed. Hence, instead of one additional forward pass, as in the RevNet and layer-wise architectures, our hybrid architecture comes with a computational cost equivalent to performing two additional forward passes during the backward pass.

4.5 Results and Discussion

We use the CIFAR10 dataset as a benchmark for our experiments. The CIFAR10 dataset is complex enough to require efficient architectures to reach high accuracy, yet small enough to enable us to rapidly iterate over different architectural designs. We start by analyzing numerical errors arising in layer-wise invertible and hybrid architectures, and outline their impact on accuracy. This analysis motivates our choice of architecture and hyperparameter. We then summarize the benefits and drawbacks of our proposed architecture in comparison to different baseline architectures.

4.5.1 Impact of Numerical stability

4.5.1.1 Layer-wise Invertible Architecture

In this section, we quantify the accumulation of numerical errors in layer-wise invertible architectures and analyze their impact on the accuracy. The architecture of these models is illustrated in Figure 4.7. We investigate the evolution of numerical errors, and their impact on accuracy, for networks of different depth and different hyper-parameter values. Figure 4.10 illustrates the degradation of the signal-to-noise ration along the layers of one such model.

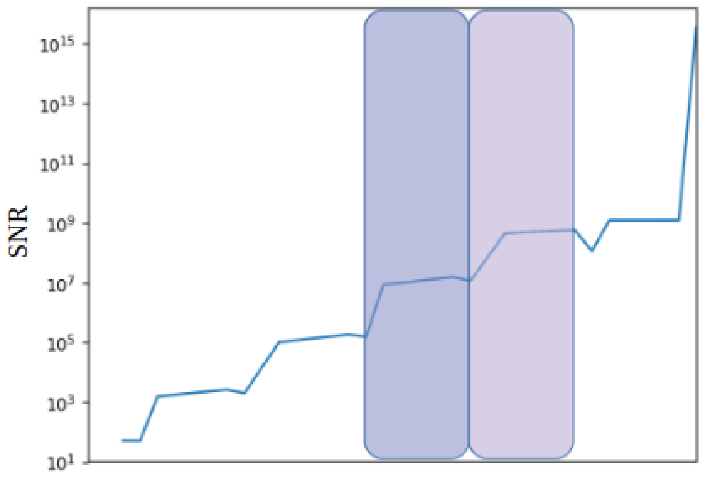

Figure 4.10: Evolution of the SNR through the layers of a layer-wise invertible model. Color boxes illustrate the span of two consecutive convolutional blocks (Convolution-normalization-activation layers). The SNR gets continuously degraded throughout each block of the network, resulting in numerical instabilities.

We found the two most impacting parameters to be the depth N of the network and the negative slope n of the activation function. Figure 4.11 shows the evolution of the numerical errors with both of these parameters.

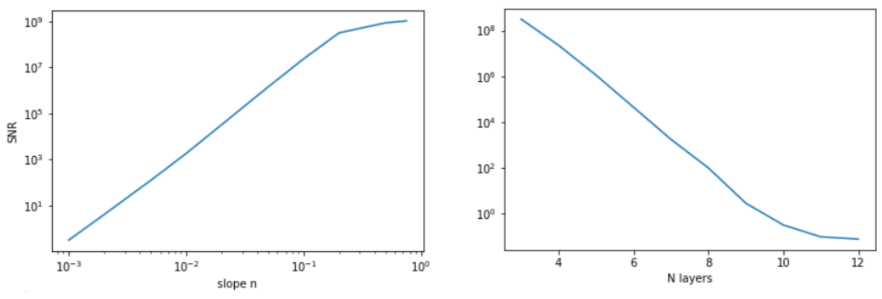

Figure 4.11: Illustration of the impact of depth (in number of layers N) and negative slope n on the numerical errors. Both figure shows the evolution of the SNR at the lowest layer of a layer-wise invertible network with increasing depth and negative slopes. The lower the SNR is, the more important numerical errors of the inverse reconstructions are. (Left): The SNR decreases exponentially with depth until it reaches an SNR value of 1. At this point, the noise is of the same scale as the signal, and no learning can happen. These results were computed with a negative slope of $n = 2$ (Right) This figure shows the evolution of the SNR with different negative slopes n for a layer-wise reversible model of depth 3. On a log-log scale, this figure shows an almost linear relationship between negative slope and SNR. It is impressive that with only three layer depth, a negative slope of $n = 10^{-3}$ reaches a SNR superior to 1. With such parameterization, even the most shallow models are not capable of learning.

Next, we investigate the impact of numerical errors on the accuracy. In order to isolate the impact of the numerical errors, we compare the accuracy reached by the same architecture with and without inverse reconstruction of the hidden layers activations. Without reconstruction, the hidden activation values are stored along the forward pass and the gradient updates are computed from the true, noiseless activation values, so that the only difference between both settings is the noise introduced by the inverse reconstructions.

In Figure 4.12, we compare the evolution of the accuracy in both settings for different depth and negative slopes. For small depths (or high negative slopes), in which the numerical errors are minimum, both models yield similar accuracy. However, as the numerical errors grow, the accuracy of the model goes down, while the accuracy of the ideal baseline keeps increasing, which can be seen with

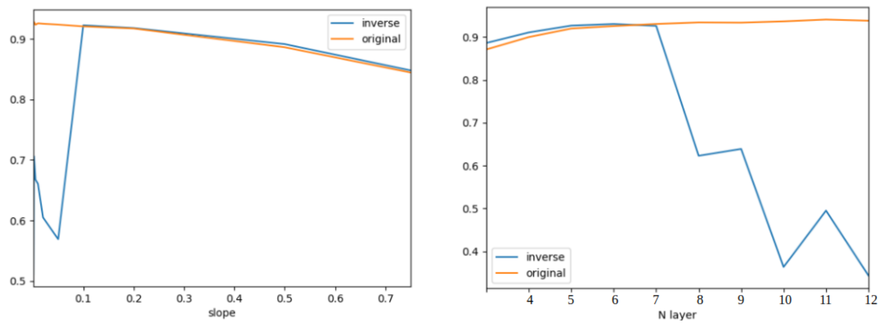

Figure 4.12: Impact of the numerical errors on the accuracy of layer-wise invertible models. (Left): Evolution of a 6-layer model accuracy with and without inverse reconstructions with the negative slope. Without reconstruction, the model accuracy benefits from smaller negative slopes. With inverse reconstructions, the model similarly benefits from smaller negative slopes as n decreases from 1 to 0.1. For smaller negative slopes, however, the accuracy sharply decreases toward lower values due to numerical errors. (Right) Evolution of the accuracy with depth for a negative slope $n = 0.2$ with and without inverse reconstructions. Without reconstruction, the model accuracy benefits from depth. With inverse reconstructions, the model similarly benefits from depth as the number of layers grow from 3 to 7. For $N > 7$, however, the accuracy sharply decreases toward lower values due to numerical errors.

both depth and negative slopes. This loss in accuracy is the direct result of numerical errors, which prevent the model from converging to higher accuracies.

4.5.1.2 Hybrid Invertible Architecture

In section 4.2, we introduced a hybrid architecture, illustrated in Figure 4.8, to prevent the impact of numerical errors on accuracy. Figure 4.13 shows the propagation of the signal to noise ratio through the layers of such hybrid architecture. As can be seen in this figure, the hybrid architecture is much more robust to numerical errors as activations are propagated from one reversible block to the other using the reversible block inverse computations instead of layer-wise inversions.

Figure 4.14 shows the evolution of the SNR with increasing depth N and for different values of negative slope n. This figure shows a much more stable evolution of the signal to noise ratio than the layer-wise architecture.

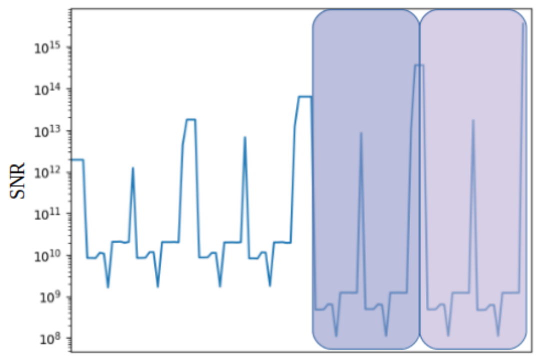

Figure 4.13: Evolution of the SNR through the layers of a hybrid architecture model. The span of two consecutive reversible blocks are shown with color boxes. Within reversible blocks, the SNR quickly degrades due to the numerical errors introduced by invertible layers. However, the signal propagated to the input of each reversible block is recomputed using the reversible block inverse, which is much more stable. Hence, we can see a sharp decline of the SNR within the reversible blocks, but the SNR almost raises back to its original level at the input of each reversible block.

Figure 4.15 compares the evolution of the accuracy reached by this hybrid architecture with noisy activations and noiseless ideal activations as depth and negative slope increase. The negative impacts of numerical errors observed in the layer-wise architecture are gone, confirming that the numerical stability brought by the hybrid architecture effectively stabilizes training.

4.5.2 Model comparison

Table 1 summarizes our main results. In this table, we compare architectures with different patterns of reversibility. To allow for a fair comparison, we have tweaked each architecture to keep the number of parameters as close as possible, with the notable exception of the i-RevNet architecture. The i-Revnet pooling scheme enforces a quadratic growth of its parameters with each level of pooling. In order to keep the number of parameters of the i-RevNet close to the other baselines, we would have to drastically reduce the number of channels of lower layers, which we found yield poor performance. Furthermore, it should be noted that the i-RevNet architecture we present slightly differs from the original i-Revnet model

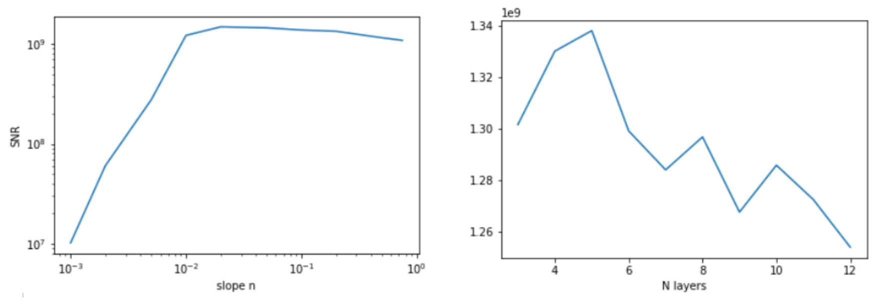

Figure 4.14: Illustration of the impact of depth (in number of layers N) and negative slope n on the numerical errors. Both figure shows the evolution of the SNR at the lowest layer of our hybrid architecture with increasing depth and negative slopes. Our hybrid architecture greatly reduce the impact of both depth and negative slopes on the numerical errors

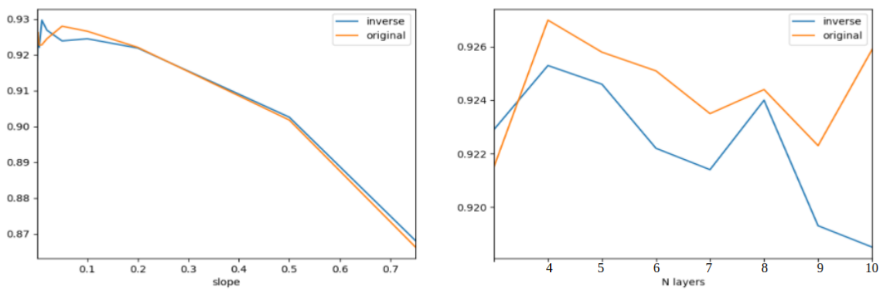

Figure 4.15: Impact of the numerical errors on the accuracy of layer-wise invertible models. Our proposed hybrid architecture greatly stabilizes the numerical errors, which results in smaller effects of the depth and negative slope on accuracy.

as our implementation uses RevNet-like reversible modules with one module per channel split for similarity with the other architecture we evaluate instead of the single module used in the original architecture.

All models were trained for 50 epochs of stochastic gradient descent with cyclical learning rate and momentum [98] with minimal image augmentation.

The parameters of our proposed architecture are given in Table 1. This architecture was selected as the best performing architecture from an extensive architecture search on a constrained weight budget. Compared to the original ResNet architecture, our model drastically cuts the memory cost of training. These drastic memory cuts come at the cost of a small degradation in accuracy.

Model	Accuracy	#Params	Channels	Pooling	M_θ	$M'_z + M'_g$	$\mathcal{M}$
Resnet	94.7%	$3.1M$	$32 - 64 - 128 - 256$	Max Pooling	$12.5M$	1928	$1.01G$
RevNet	94.5%	$3.1M$	$40 - 80 - 256 - 320$	Max Pooling	$12.7M$	640	$348M$
i-RevNet	93.8%	$42.8M$	$32 - 128 - 512 - 2048$	$\mathcal{P}_c - \mathcal{P}_c - \mathcal{P}_c$	$171M$	640	$500M$
Ours	93.3%	$3.7M$	$32 - 128 - 512 - 512$	$[\mathcal{P}_c, \mathcal{P}_c, \mathcal{P}_b]$	$14.8M$	352	$200M$

Table 4.1: Summary of architectures with different levels of reversibility

GPU	Accuracy	Time
GTX750	93.3%	$67min.$
GTX 1080Ti	93.3%	$35min.$

Table 4.2: Training statistics on different hardware

Furthermore, our hybrid architecture requires the computational equivalent of two additional forward passes within each backward pass. The computational complexity, however, remains reasonable: In Table 2, we compare the time of training our proposed architecture to 93.3% on a high-end Nvidia GTX 1080Ti and a low-end Nvidia GTX750. The GTX750 only has 1GB of VRAM, which results in roughly 400MB of available memory after the initialization of various frameworks. Training a vanilla ResNet with large batch sizes on such limited memory resources is impractical, while our architecture allows for efficient training.

4.5.3 ZSL Benchmark

Finally, we integrate our proposed visual module to the ZSL pipeline described in Figure 1.1 and evaluate the accuracy of the our model on the ZSL benchmark proposed in Chapter 3. Table 3 summarizes our results for different core ZSL modules

Table 4.3: Evaluation on the proposed benchmark. Accuracy in the generalized ZSL setting are reported as harmonic means over training and test accuracy following [4]

Model	ZSL		G-ZSL	
	@1	@5	@1	@5
CONSE [27]	8.7	18.7	0.15	16.7
DEVISE [3]	9.8	26.8	**6.8**	23.3
ESZSL [16]	12.6	29.0	3.5	22.1
GCN-6 [65]	8.9	23.1	3.71	19.2
GCN-2 [5]	12.9	31.8	3.82	**26.9**
ADGPM [5]	**13.10**	**32.03**	4.06	26.2

4.6 Chapter conclusion

Convolutional Neural Networks form the backbone of modern computer vision systems. However, the accuracy of these models come at the cost of resource intensive training and inference procedures. While tremendous efforts have been put into the optimization of the inference step on resource-limited device, relatively little work have focused on algorithmic solutions for limited resource training. In this paper, we have presented an architecture able to yield high accuracy classifications within very tight memory constraints. We highlighted several potential applications of memory-efficient training procedures, such as on-device training, and illustrated the efficiency of our approach by training a CNN to 93.3% accuracy on a low-end GPU with only 1GB of memory.

Chapter 5

Conclusions

ZSL has the potential to be of great impact, for both practical applications and theoretical investigation. Despite its great promises and after a decade of active research, the accuracy of ZSL models on the standard benchmark remain too low for practical applications.

In this thesis, we questioned some fundamental elements of Zero-Shot recognition of generic objects. We first proposed a definition of the goal of ZSL: combinatorial generalization accross classes by composition of visual features. We showed that the current benchmark was ill-suited to track progress towards this goal, and introduced the notion of structural bias to formalize and quantify this claim. Structural bias in the standard benchmark has influenced the development of ZSL models by encouraging models that maximally exploit similarity-based matching in semantic space.

We have discussed the relevance and limitations of word embeddings for ZSL and proposed to automatically collect graph and text documents as alternative descriptions of visual concepts. Finally, we discussed the role of texture and shape bias in ZSL and proposed a new architecture for memory constrained extraction of visual features.

We questioned the relevance and limitations of word embeddings as semantic representation of visual classes. We highlight several important limitations of defining large scale visual concepts with words and quantify the impact of these limitations on the accuracy of ZSL systems. Instead of words, we proposed that visual classes be defined by explicit descriptions in the form of text documents and structured data, and automated the acquisition of such descriptions.

We have investigated the application of invertible transformations to reduce the memory usage of training large CNN. We analyzed the memory bottlenecks in training existing revertible networks and propose a layer-wise invertible architecture to alleviate these bottleneck. We characterized the memory consumption and numerical errors arising in long chain of invertible transformations and, based on our analysis, propose an architecture with minimal memory footprint.

Finally, we believe that a deeper discussion on the goals and definition of ZSL is still very much needed. There is a risk in developing complex models to address poorly characterized problems: Mathematical complexity can act as a smokescreen of complexity that obfuscates the real problems and key challenges behind ZSL. We believe that practical considerations grounded in common sense are still very much needed at this stage of ZSL research. Taken together, we hope that the contributions presented in this thesis provide a solid base for the development of future ZSL research.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. ii, 1, 33

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. ii, 1, 33, 35

[3] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013. xvii, 10, 28, 29, 33, 35, 50, 54, 55, 91

[4] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591, 2017. xvii, 10, 28, 29, 35, 50, 54, 91

[5] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P Xing. Rethinking knowledge graph propagation for zero-shot learning. *arXiv preprint arXiv:1805.11724*, 2018. xvii, 29, 35, 50, 54, 91

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[12] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015. 1

[13] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):652–663, 2016. 1

[14] Noam Chomsky. *Aspects of the Theory of Syntax*, volume 11. MIT press, 2014. 3

[15] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 3

[16] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015. 5, 29, 35, 50, 54, 91

[17] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Semantic embeddings of generic objects for zero-shot learning. *EURASIP Journal on Image and Video Processing*, 13, 2019. 9

[18] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Visually grounded word embeddings for zero-shot learning of visual categories. *Workshop on Computer Vision and Image Media*, 2018. 9

[19] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Investigation of the correlations between cnn visual features and word embeddingsaining. *Second Workshop on Closing the Loop Between Vision and Language, International Conference on Computer Vision, Deceber 2017, Venice*, 2017. 9

[20] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Zero-shot learning using dictionary definitions. In *Proceedings of the International Workshop on Frontiers of Computer Vision.*, 2018. 9

[21] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Knowledge graph embeddings for zero-shot learning. *National Symposium on Image Processing and Understanding*, 2018. 9

[22] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Semantic web and zero-shot learning of large scale visual classes. In *Proceedings of the First Workshop on Symbolic Neural Learning*, 2017. 9

[23] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 9

[24] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009. 9, 34

[25] Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3174–3183, 2017. 10, 28

[26] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016. 10, 28, 29, 35, 50

[27] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. 10, 28, 29, 35, 50, 54, 55, 91

[28] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 10, 14

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 10, 14

[30] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, pages 6341–6350, 2017. 10, 22, 23, 24

[31] Marcus Rohrbach, Michael Stark, György Szarvas, Iryna Gurevych, and Bernt Schiele. What helps where–and why? semantic relatedness for knowledge transfer. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 910–917. IEEE, 2010. 11

[32] Thomas Mensink, Efstratios Gavves, and Cees GM Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2441–2448. IEEE, 2014. 11

[33] Tanmoy Mukherjee and Timothy Hospedales. Gaussian visual-linguistic embedding for zero-shot recognition. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 912–918, 2016. 11

[34] Qian Wang and Ke Chen. Alternative semantic representations for zero-shot human action recognition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 87–102. Springer, 2017. 11

[35] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 2927–2936. IEEE, 2015. 11, 29

[36] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77, 2016. 11, 29, 35, 42, 50

[37] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438, 2016. 11, 29

[38] Yutaro Shigeto, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer, 2015. 12

[39] Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*, 2014. 12

[40] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014. 14

[41] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247, 2014. 14

[42] Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. Bag of tricks for efficient text classification. *EACL 2017*, page 427, 2017. 14

[43] Liyang Yu. Linked open data. In *A Developers Guide to the Semantic Web*, pages 409–466. Springer, 2011. 21

[44] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012. 21

[45] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning.* MIT press, 2007. 22

[46] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017. 22

[47] Benjamin Paul Chamberlain, James R Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. 22

[48] Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. Representation tradeoffs for hyperbolic embeddings. 2018. 22

[49] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, 2017. 23

[50] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. 2015. 23

[51] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013. 23

[52] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. 2017. 23

[53] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, 2017. 26

[54] Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30, 2016. 26

[55] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015. 26

[56] Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, 2016. 26, 27

[57] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014. 26

[58] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. Technical report, 2017. 27

[59] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004. 27

[60] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003. 27

[61] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016. 27

[62] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1870–1879, 2017. 27

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 28

[64] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013. 29, 50

[65] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6857–6866, 2018. 29, 35, 50, 54, 91

[66] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. On zero-shot recognition of generic objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, 2019. 33

[67] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 33

[68] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008. 33

[69] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2751–2758. IEEE, 2012. 34

[70] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010. 34

[71] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 35

[72] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011. 35, 48

[73] Allan Jabri, Armand Joulin, and Laurens van der Maaten. Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer, 2016. 35

[74] Kushal Kafle and Christopher Kanan. An analysis of visual question answering algorithms. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 1983–1991. IEEE, 2017. 35

[75] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017. 35

[76] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Reversible designs for extreme memory cost reduction of cnn training. *EURASIP Journal on Image and Video Processing*, Under review. 57

[77] Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018. 58

[78] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018. 58, 59

[79] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, pages 2214–2224, 2017. 59, 60, 80

[80] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018. 59, 60, 80

[81] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 59

[82] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 59

[83] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018. 60

[84] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015. 60

[85] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018. 60

[86] Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. *arXiv preprint arXiv:1811.00401*, 2018. 60

[87] Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018. 60

[88] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5639–5647, 2018. 60

[89] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 61

[90] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 61

[91] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 61

[92] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 61

[93] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017. 61

[94] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. 61

[95] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*, 2018. 61

[96] James Martens and Ilya Sutskever. Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012. 61

[97] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 62

[98] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017. 89

Acknowledgements

I would like to thank my supervisors, Emeritus Professor Yasuo Ariki and Professor Tetsuya Takiguchi at Kobe University, who have given me helpful advice and continued support during my research.

I am deeply grateful to my dear friend Quentin for visiting me and spending time conducting research together. The time he spent with me have been the best period of my Ph.D.

Lastly, I would like to thank my family: My partner Kim'chan Kurara, for everything she has done to support me during my time as a Ph.D student and for the good energy she gave me everyday. My daughter Kellen for being amazing. I would like to thank my parents for everything they have given me. Also, my brother and sister are rockstar inspirations. The meer though of their presence is my biggest strength. Big love.

BibTeX Citation for This Thesis

```
@article   {TristanThesis2019,
             title={{Zero Shot Recognition of Generic Objects
             journal={Doctoral Thesis},
             author={Tristan Hascoet},
             institution={Kobe University},
             month={July},
             year={2019}
           }
```

Doctor Thesis, Kobe University
“Zero Shot Recognition of Generic Objects”, 132 pages
Submitted on July, 19, 2019.
The date of publication is printed in the cover of repository version published in
Kobe University Repository Kernel.