Deep learning with 3D and label geometry

Liao, S.

Publication date
2021
Document Version
Final published version

Deep learning with 3D and label geometry

Shuai Liao

Deep learning with 3D and label geometry

Shuai Liao

Deep learning with 3D and label geometry

This book was typeset by the author using LaTeX 2_ε.

Deep learning with 3D and label geometry

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het college voor promoties
ingestelde commissie,
in het openbaar te verdedigen in de Aula
op dinsdag 22 september 2021 te 11:00 uur

door

Shuai LIAO

geboren te Sichuan, China

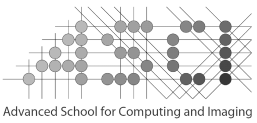
Promotiecommissie

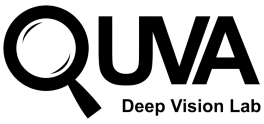
Promotor:	Prof. dr. C. G. M. Snoek	Universiteit van Amsterdam
Co-promotor:	Dr. E. Gavves	Universiteit van Amsterdam
Overige leden:	Prof. dr. ir. C. Sánchez Gutiérrez	Universiteit van Amsterdam
	Prof. dr. ir. A. W. M. Smeulders	Universiteit van Amsterdam
	Prof. dr. Th. Gevers	Universiteit van Amsterdam
	Prof. dr. R. C. Veltkamp	Universiteit Utrecht
	Dr. X. Li	Renmin University, China
	Dr. A. Ghodrati	Qualcomm AI Research

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

UNIVERSITEIT VAN AMSTERDAM

This work was carried out in the ASCI graduate school, dissertation series number 423, at the Video & Image Sense lab, and the QUVA lab of the University of Amsterdam.





Contents

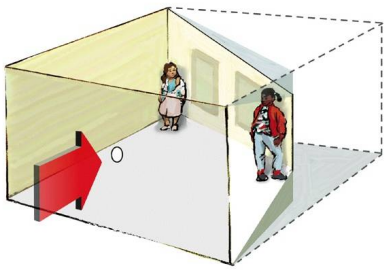
Chapter 1

Introduction

The breakthrough that revolutionized modern computer vision was made at the University of Toronto in 2012 [83]. Their deep learning architecture, AlexNet, achieved tremendous success in modelling the large-scale ImageNet computer vision challenge [134], in which an algorithm is asked to classify millions of images into a thousand classes. This image classification model, known as a Deep Convolutional Neural Network (DCNN), is loosely inspired by the billions of interconnected neurons in our brain. Typically, a deep convolutional neural network is built upon a stack of convolutional layers, with each layer containing hundreds of thousands of functional connections, *i.e*. the artificial neurons. The visual representations are processed and transformed layer by layer, resembling in a relaxed way the function of the neurons in the visual cortex of the brain. In comparison to the traditional hand-crafted features, *e.g*. [144, 14, 104, 22, 39], the DCNN can be trained from scratch in an end-to-end fashion through the gradient back-propagation. This relieves us from the reliance on expertise required when designing hand-crafted feature descriptors. Today, going deeper [143], wider [176] and having more connections [66] are the key characteristics of the newly emerged deep neural network architectures [70, 33, 147, 41]. With the increasing capacity of deep neural networks, computers are reaching human-level -or even superhuman- accuracy in image classification [58]. The similar successes are also achieved in object detection [47, 132, 59, 130, 101], action recognition [142, 150, 32, 159, 15, 161, 174], creating artistic or photo realistic [181, 73] images, and many more applications.

Despite the recent progress in using deep learning to solve computer vision problems, having a fine-grained understanding of an image remains challenging. Often, such understanding of an image is two-fold: the visual understanding and the semantic understanding. The former strives to understand intrinsic properties of the object in the image, *e.g*. the 2D visual appearance, the 3D shape, the 3D position and the 3D pose *etc*., whereas the latter aims at associating the diverse objects with certain semantics, *e.g*. a category name of an object [47, 132, 59, 130, 101], an action [142, 150, 32, 159, 15, 161, 174] or an attribute [135, 99, 158, 135]. All of these form the basis of an in-depth understanding of images that we wish a machine to have.

(A) **The Ames room***



(B) **The underlying 3D geometry**†

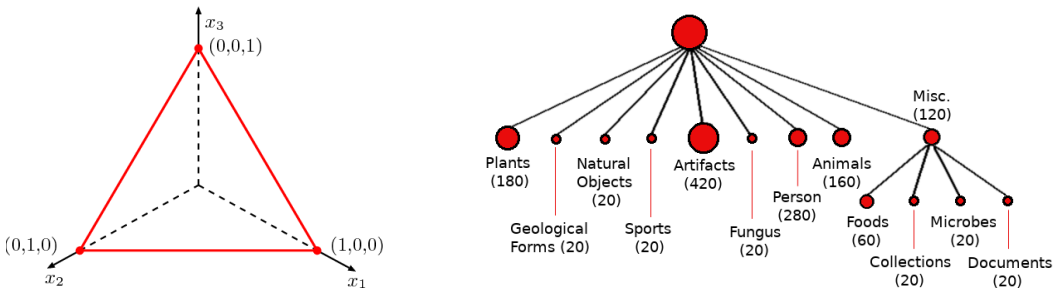
FIGURE 1.1: (A) **The Ames room** creates an illusion that the man on the right is much higher and bigger than the man on the left. (B) **The underlying 3D geometry** of Ames room shows that the room is not square. When looking at the room from a certain viewpoint, our vision system falsely perceives its 3D geometry. Whereas geometric distortion of the room is the cause of the illusion, the implicit assumption of a square room is the root of the perceptive malfunction. This illusion demonstrates the strong bias that the human brain has in exploiting 3D geometry to interpret visual information.

Today's default architectures of deep convolutional networks have already shown a remarkable ability in capturing the visual appearances of images in the 2D domain, and mapping visual content to one specific semantic class thereafter (*e.g.* image classification, action recognition). However, research on fine-grained image understanding, such as inferring the intrinsic 3D information and more structured semantics, is less explored. In this thesis, we contribute to the two aspects by studying how geometry can be used to better understand images. Next, we motivate our angles in looking at the problems of visual understanding and semantic understanding of an image.

1.1 3D geometry and visual understanding

First, let us take a look at Fig. 1.1-(A). The person on the right appears much taller and bigger than the person on the left. However, they both have a similar size in reality. This room, known as the *Ames room*, constitutes a visual illusion by its unique design. The magic of this room is that it is geometrically distorted. Fig. 1.1-(B) shows a sketch of the real geometry of this room. When looking at it from a certain viewpoint, the heights at the left corner and right corner of the room appear to be the same. However, in reality, the room is not square. The result of the illusion is that the person on the right appears to be much taller than the person on the left, while in reality, the person to

* Photo credit Zach King; †Photo credit Dean Odell .



(A) **The probabilistic simplex** (B) **A class hierarchy**

FIGURE 1.2: Two example of *label geometries* in image classification. (A) **The probabilistic simplex** plane [166] for a three-classes classification problem, where only one label can be assigned to a sample (*i.e.* categorical distribution). The constraint on the summation of probabilities in being each of the classes equals to constitute such label geometry. (B) **A class hierarchy** enforces a more complex label geometry for probabilistic model: the probability of an image being an internal node on the hierarchy should equal the summation of probabilities from all its children nodes. Label geometries help to build semantically meaningful image classification models.

the right is simply closer to the camera. This illusion demonstrates the strong bias that the human brain has in exploiting 3D geometry to interpret visual information.

Although the above is an example of a failure case of our vision system, it reflects the importance of 3D geometry in our visual perception, albeit we only take 2D signals as inputs. Analogously, we posit that it is important for a computer to explore the 3D geometry when processing 2D visual contents from images. In this thesis, we study how to algorithmically derive the 3D geometry information from a 2D image, and reversely make an effort to explain a 2D image with 3D geometry.

1.2 Label geometry and semantic understanding

Often, it is the case that we need to communicate with each other in more detail beyond just describing the visual appearances of objects. This leads to another level of image understanding - the semantic understanding - linking what we see with what we call it. In this thesis, the semantics of an image mainly refers to the abstraction of visual contents in the form of nouns, or categories. We confine our discussion to approaches that associate semantics with images in the scope of image classification. Just like the role of the 3D geometry in visual understanding, semantic understanding of images also calls for geometry.

Typically, image classification is solved with a probabilistic model, which outputs the probabilities of the input image depicting each of the classes. The way probabilities are organized to the classes can be viewed from a geometrical perspective as well. For example, let us consider a three-class classification problem with only one correct label. The probability of a sample belonging to each of the classes, denoted as $\{p_1, p_2, p_3\}$, follows a categorical distribution, and, therefore, the probabilities must sum up to one, *i.e.* $p_1 + p_2 + p_3 = 1$. This corresponds geometrically to a triangular plane, see Fig. 1.2-(A), known as a simplex [166]. Another example of geometry and semantics is image classification with more complex class structures, such as class hierarchies, see Fig. 1.2-(B). In this case, a probabilistic model must ensure that the probability of an image that belongs to an internal node on the hierarchy is equal to the sum of probabilities of all children nodes below it. All these constraints constitute a complex probabilistic geometry that makes the predictions on the hierarchy semantically meaningful.

Borrowing the terminology from supervised image classification, where the semantic category names are called "labels", we refer to the aforementioned geometries in the probability space as "label geometry". Whereas previous works mainly focus on label geometry in one-*vs*.-rest classification, in this thesis, we will discuss how to model label geometry when multiple labels or none of the labels are present in an image. Interestingly, we will find that the label geometry is also possible to be learned from, rather than be imposed to, an image classification model.

1.3 Research Questions

This thesis strives to answer:

How to better utilize geometry for better image understanding?

Visual content and semantic meaning are two crucial perspectives in image understanding. We find that although geometry plays an important role in both perspectives, it is not fully explored in the context of deep learning. Specifically, these directions that await to be explored further include, but are not limited to, 1) the 3D geometry such as the shape and the pose of an object, the viewpoint of a camera *etc*., and 2) the label geometry that links individual visual instances in images to linguistics. Thereupon, we initiate our exploration of the 3D geometry in visual image understanding and the label geometry in semantic image understanding. In the following, we briefly introduce the four corresponding chapters build upon their research questions.

Part I. Deep learning with 3D geometry

In the first part of this thesis, we start from the role of 3D geometry in visual understanding. When interpreting a scene or an object, our brain usually resorts to its life experiences of 3D geometry as prior knowledge, instead of explicitly reconstructing the 3D geometry from scratch. The same idea is also reflected by early works that rely on certain 3D geometric templates to parse 2D images [117, 80, 69, 119, 9, 10, 63]. Unlike the variety of prior knowledge of the 3D geometry that humans build throughout life, the number of 3D shapes or templates used by models is often rather limited.

In recent years, however, it is becoming increasingly easier to access the prior knowledge of 3D shapes from large-scale 3D shape databases. For example, ShapeNet [17] collects over three million 3D CAD models from the internet and organizes them into a taxonomy. With more than 3K categories, most of the commonly seen objects like "car", "airplane" or "desk" are well covered by such a 3D database. This offers computers an opportunity to exploit them and build their own prior knowledge of the 3D geometry for image parsing. The way how to build this prior knowledge is open and certain questions stand along the way. First, given a large enough 3D database, how can one find the right 3D shape to match with objects in a 2D image? Second, the 3D shapes in such database either usually miss texture or their textures miss realism, thus being very different from the variety of appearances and textures an object can have in reality and in images. Thereupon, our first research question is,

How to search and match texture-free 3D shapes to a 2D image?

We address this research question in Chapter 2. To achieve this goal, we first need to design an algorithm to automatically retrieve the right 3D shape from the database based on the 2D appearance of an object in the image, regardless of its texture. This can be done by training a deep classification network, which takes an RGB image of an object as input and outputs the most likely type of 3D shape. Second, given the retrieved texture-free 3D shape, we need to find a correct location and a pose such that the rendered 3D shape aligns with the object in a 2D image. Matching rendered views of 3D shapes to RGB images is challenging because, 1) 3D shapes can not always be perfectly matched to the image queries, 2) there is a great domain difference between rendered and RGB images, and 3) estimating the object scale is inherently ambiguous in images taken from uncalibrated cameras, as the size of an object in the pixel space depends not only on the physical size of the object but also the distance from the camera. To address these challenges, we propose a deeply learned matching function that compares a rendered view of 3D shape with the object in a 2D query image. Through multiple such

comparisons, the object location and pose in the 2D image can be decided by extracting the rendering parameters of the best matching rendered view.

In the next chapter, we explore the 3D geometry in a 2D image in an opposite direction. We try to derive the 3D geometry out of a 2D image, rather than use the 3D geometry as prior knowledge to parse a 2D image. Early works focused on the 3D shape and structure reconstruction from a sequence of images [152, 136, 167, 122, 16]. In the pursue of a finer understanding, we rather aim at recovering a set of 3D geometric attributes that are even more elementary than 3D shapes, such as the direction of the normal vector on a surface, the orientation angles of a camera, and the rotation of an object. The advantage is that it is possible to estimate these elementary attributes from single images. Furthermore, the inference speed is usually much faster than algorithms that trying to recover the exact 3D shape of an object, which is critical for applications such as robotics and autonomous driving. This direction was pioneered by Lawrence Roberts who described the process of deriving the 3D position and orientation of a simple planar-surfaced shape from line drawing as early as 1963 [133]. More recently, complex objects such as chairs and cars are studied in a cluttered scene under the deep learning framework [145, 151, 160, 126, 90, 177]. However, we find that although most 3D targets are continuous in nature, modern deep learning-based solutions tend to output discrete predictions. A natural way to model these 3D targets would be with regression. However, most works [145, 151, 44] prefer to approach the problem as a classification one, discretizing the continuous targets to a set of discrete outputs and modeling them with deep neural networks. The main reason for this paradox is that historically, it has been shown that classification lends itself to a more stable and consistent optimization procedure, thus yielding better final predictions. Still, with classification, the models can only predict a limited number -and not the continuous spectrum- of possible targets. Thereupon, the second research question becomes:

Is it possible to train deep neural networks that output continuous estimations of viewpoints, rotations, and surface normal from a 2D image in a reliable and accurate manner?

This research question is considered in Chapter 3. A reason to explain the difference in the stability when training classification and regression models is that in classification the output is naturally contained within a closed geometry, *i.e.* the probability n-simplex [11]. Defined by the popular softmax activation function, this closed geometry not only restricts the learning space but also helps to further stabilize the gradient. In contrast, regular regression does not feature such a closed geometry in its output, possibly leading to unstable training and convergence to sub-optimal local minima. Starting from this insight we revisit regression in convolutional neural networks. We observe many problems with continuous output in computer vision are

naturally contained in closed geometrical manifolds, like the Euler angles in viewpoint estimation or the normals in surface normal estimation. A natural framework for posing such continuous output problems is n-spheres, which are naturally closed geometric manifolds defined in the $R^{(n+1)}$ space. By introducing a spherical exponential mapping on n-spheres at the regression output, we obtain well-behaved gradients, leading to stable training. We show how our spherical regression can be utilized for predicting several challenging 3D targets, specifically viewpoint estimation, surface normal estimation, and 3D rotation estimation.

Part II. Deep learning with label geometry

In the second part, we will focus on the semantic understanding of 2D images in the context of geometry. Image classification, which is at the heart of semantic image understanding, supports a wide range of applications such as image retrieval, tagging, and recommendation. In general, the goal of image classification is to learn a function that maps a 2D input image to a set of predefined class labels. These class labels are the linguistic abstractions of images, which can be the name of an object that appears in the image or a tag that is associated with the image by social media users. In a probabilistic model, the probabilities of each of the labels per image are geometrically connected by, for example, the assumption of the number of labels a sample has or the prior knowledge of a class hierarchy. Therefore, studying the label geometry potentially helps to build better probabilistic models for image classification.

Instead of assuming that an image will always contain one and one only label, we note that an image may be associated with a single label, multiple labels, or even no label given a vocabulary. Present-day methods resort to different probabilistic models for each of these problems: one-*vs*.-rest classification [83, 143, 134, 58], multi-label classification [21, 91, 128, 49] or out-of-distribution classification [61, 92, 88]. We note that the three problems are related in the sense that they all refer to a similar modeling task, that of automatically inferring a class label given an image, the difference being in the number of labels the image is supposed to contain. In a probabilistic model, knowing the numbers of labels in an image introduces similar label geometries, possibly allowing for modelling all three problems in a unified manner. This prior knowledge on the number of labels is typically freely accessible and can learned from the training set. Therefore, the third research question is:

How to leverage the label geometry to unify image classifiers?

This research question is considered in Chapter 4. Typically, one-*vs*.-rest classification is solved by a softmax classifier, which models a categorical

distribution. In contrast, multi-label classification is solved by an ensemble of binary classifiers, which models a set of independent Bernoulli distributions. We observe the only difference between binary and softmax classifiers is their normalization functions, which capture different label geometries. Specifically, while the binary classifier self-normalizes its scores, the softmax classifier combines the scores from all classes before normalization. Based on this observation we introduce a normalization function that is learnable, constant, and shared between classes and data points. By doing so, we arrive at a new type of binary classifier that we coin quasibinary classifier. We show in a variety of image classification settings, and on several datasets, that quasibinary classifiers are considerably better in classification settings where regular binary and softmax classifiers suffer, including zero-label and multi-label classification. What is more, we find the quasibinary classifiers yield well-calibrated probabilities allowing for direct and reliable comparisons, not only between classes but also between data points.

In the last chapter, we consider a more complex label geometry: the taxonomy of categories. Most existing image classification models are by design "flat" [143, 66, 132, 59, 130, 101, 70, 33], meaning that each class is treated equally, where each class is at the finest granularity and all classes are mutually exclusive. When classifying images, however, we are not necessarily interested in the finest of categorizations if this increases the chances of a misclassification. For instance, we are often content with recognizing an "eagle" or even "bird" instead of a "Montagu's Harrier", rather than confuse that bird's image with an "airplane". To accommodate this, a simple solution is to make predictions in bundles, where bundles contain confused classes without further differentiation. As we cannot know in advance which classes get confused or even what is the optimal way to organize them in bundles, a common way is to group classes in terms of a hierarchy. When confusion emerges between the most fine-grained leaf classes, predicting the internal node (a bundle) on the higher level of a hierarchy is able to achieve an arbitrary high accuracy, at the cost of sacrificing specificity [26]. However, creating a large scale of class hierarchy such as Wordnet [154] or iNaturalist [112] typically involves a huge amount of domain experts' efforts. Consequently, the previous approaches in balancing accuracy with specificity are only applicable to a few datasets with a semantic hierarchy available. Therefore, we ask our last research question:

How to infer the label geometry from image classification to balance accuracy* vs. *specificity?

This research question is considered in Chapter 5. To learn such a label geometry, we introduce a new embedding layer able to learn by discriminative training class-prototypes from which a visual hierarchy is extracted.

	Geometry → Image	Image → Geometry
3D Geometry (Part I)	Chapter 2 *Matching 3D to 2D*	Chapter 3 *Spherical Regression*
Label Geometry (Part II)	Chapter 4 *Quasibinary Classifiers*	Chapter 5 *Vec2Bundle*

FIGURE 1.3: **A road map of the four research topics in this thesis**.

We refer to this embedding as Vec2Bundle. Further, by introducing a negation rule in deriving probabilities on the hierarchy, we are the first to enable a trade-off between accuracy and specificity on multi-label hierarchical classification, wherein previous approaches were infeasible due to the exclusiveness of leaf nodes. We validate the effectiveness of Vec2Bundle key components with ablation experiments and compare with the state-of-the-art in balancing accuracy and specificity for both single-label and multi-label image classification. Interestingly, it appears that Vec2Bundle can capture semantics without being explicitly instructed to do so.

To summarize, in this thesis we research visual image understanding and semantic image understanding, from the perspectives of 3D geometry and label geometry. For each perspective, our research can be subdivided into two similar topics. That is, we study 1) how prior knowledge of geometry can be fit to the image to help us better interpret images (*i.e.* Geometry $\rightarrow$ Image), and reversely 2) how geometry can be derived from 2D images (*i.e.* Image $\rightarrow$ Geometry). We investigate each topic in a separate chapter, which we graphically organize in Fig. 1.3.

Co-authorship and Roles

For each chapter of this thesis we here declare the authors' contributions:

Chapter 2

Shuai Liao, Efstratios Gavves, Cees G.M. Snoek (2018). *"Searching and Matching Texture-free 3D Shapes in Images"*. In: Proceedings of the ACM International Conference on Multimedia Retrieval. [93].

- Shuai Liao — All aspects
- Efstratios Gavves — Insight and supervision

- Cees G.M. Snoek — Insight and supervision

Chapter 3

Shuai Liao, Efstratios Gavves, Cees G.M. Snoek (2019). *"Spherical regression: Learning viewpoints, surface normals and 3d rotations on* n*-spheres"*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. [94].

- S. Liao — All aspects
- E. Gavves — Insight and supervision
- C.G.M. Snoek — Insight and supervision

Chapter 4

Shuai Liao, Efstratios Gavves, ChangYong Oh, Cees G.M. Snoek (2020). *"Quasibinary Classifier for Images with Zero and Multiple Labels"*. In: International Conference on Pattern Recognition. [95].

- S. Liao — All aspects
- E. Gavves — Insight and supervision
- C. Oh — Technical advice.
- C.G.M. Snoek — Insight and supervision

Chapter 5

Shuai Liao, Amirhossein Habibian, Efstratios Gavves, Cees G. M. Snoek, Amir Ghodrati. *"Vec2Bundle: Learning Class Hierarchies to Balance Accuracy versus Specificity"*. Unpublished.

- S. Liao — All aspects
- A. Habibian — Guidance and technical advice
- E. Gavves — Insight and supervision
- C.G.M. Snoek — Insight and supervision
- A. Ghodrati — Guidance and technical advice

Part I

Deep learning with 3D geometry

Chapter 2

Searching and Matching Texture-free 3D Shapes in Images

2.1 Introduction

The goal of this chapter is to search and match the rendered view of a texture-free 3D shape to an object of interest in a 2D image. Matching shapes to image content has a long tradition in content-based image retrieval, *e.g.* [148, 89], where queries have been entered by sketching [74], by combining sketch and keywords [45], or by a provided 3D shape [2]. Inspired by these prior works, we query a dataset of 3D shapes based on an image, but we also recognize and localize an object of interest in the image and match the object to its corresponding 3D model, so as to arrive at an alignment of the 3D shape in the 2D image as precise as possible, see Fig. 2.1.

Searching and matching a 3D shape to an object in a real-world 2D image is challenging because: *(a)* finding the 3D shape that has the exact shape as the objects in the image is not always possible, even when large 3D shape libraries, *e.g.* [17], are available, *(b)* matching a rendered image from a 3D shape with a real image is known to suffer from domain shift: the real image may have different texture, lighting condition, shadow and complex natural background, and *(c)* estimating the object scale versus distance is inherently ambiguous in images taken from uncalibrated cameras, where the camera intrinsic matrix is unknown. In this chapter we study the influence of these challenges with respect to the searching and matching of 3D shapes in images.

2.1.1 Related Work

We are inspired by recent progress, mostly reported in computer vision venues, were several sophisticated methods for 3D to 2D object matching have been presented, *e.g.* [3, 52, 4]. Most methods rely their matching on exemplar classifiers, which are trained to learn an absolute relation between particular 3D model views and particular RGB appearances via texture-sensitive features,

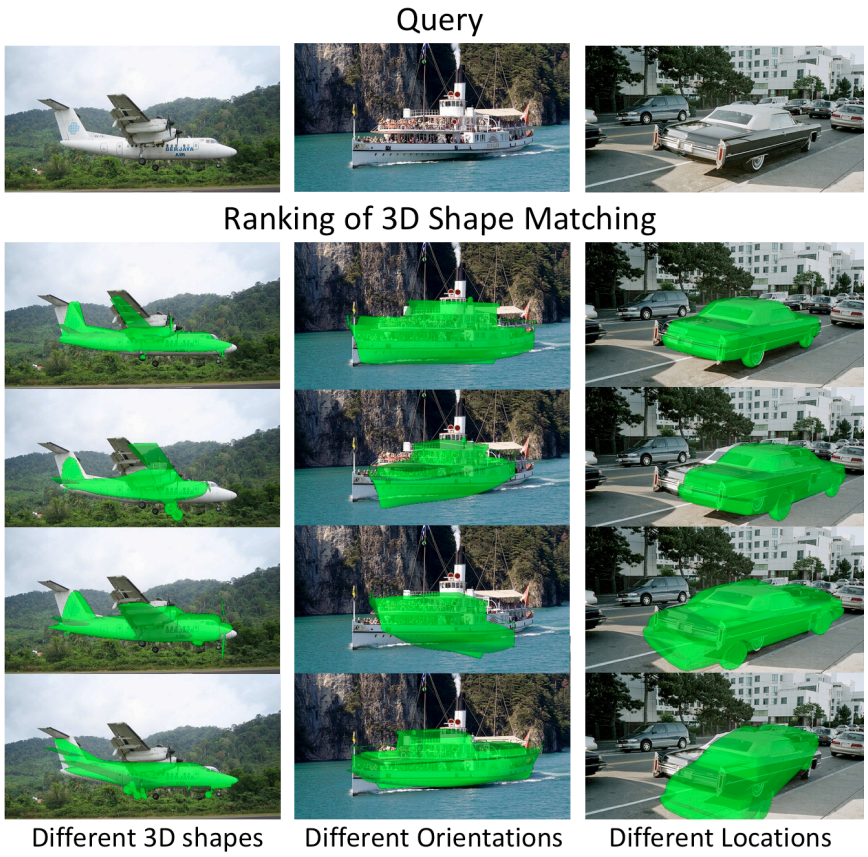


FIGURE 2.1: This chapter strives to find the rendered view of a texture-free 3D shape that best matches an object of interest in a 2D query image. This matching is challenging because the 3D shapes are not always perfect for the image query (aeroplane, boat, car), there is an obvious domain difference between the appearance of the RGB object and the texture-free rendered image, and the intrinsic camera parameters are unknown. We propose and evaluate a deeply learned matching function that attacks these challenges and can be used as a search engine that finds and matches 3D shapes to objects in 2D images.

such as HOG [3, 4]. In the seminal work of [3], for example, Aubry *et al*. rely on 3D models of chairs from image search engines to align 3D models to 2D objects. The authors propose an LDA-based exemplar classifier trained on HOG features of textured 3D models to detect the 3D model and the pose that best matches real chair images. Unfortunately, the exemplar classifiers are trained for specific models, textures, and specific poses, namely they learn a specific classifier for each 3D model, its different poses and RGB appearances separately. Hence, novel 3D models or poses cannot be accommodated without retraining, limiting the applicability of the learned model for general-purpose 3D shape retrieval.

In [96] a method is described for aligning perfectly matched 3D shapes of IKEA furniture on RGB living room images, be it they rely on parts that must be defined and annotated a priori on the perfect matched 3D shapes. In [114] 3D models are reduced to 3D cuboids, for object categories with box-like geometry, like cars, thus making inference easier. These methods learn absolute relations between *textured* 3D models and the respective 2D objects available at training. Having textured 3D models can be beneficial, *e.g.* if the texture of the object in the 2D image happens to be similar. But it can also limit the algorithm, by biasing it to retrieving 3D models that only share similar textures regardless of apparent differences in shape geometry. Even with large-scale 3D shape datasets such as [17], finding similar textured 3D models that match objects as they appear in images in the wild is still limited by the stored variation in the dataset. What is more, the texture quality and style is not necessarily consistent, as they are typically designed by different 3D artists. Consequently, texture-based methods are constrained to textures already observed, and they are unable, nor intended, to transfer their knowledge to new 3D models or poses at test time. An alternative [4] is to rely on depth RGB images to learn the 3D-to-2D matching function, limiting, however, the applicability of the algorithm to RGB-D cameras only. Unlike these approaches, our focus is a search engine of the best matching 3D shapes given an object in a 2D images. Thus, we are also interested in transferability, even for unseen 3D shapes. In this work we focus on texture-free 3D shapes and standard RGB images as queries.

To alleviate the dependence on texture appearance, Choy *et al*. [20] first render textured 3D models into background-free images from a set of discretized viewpoints. Then, they compute synthesized detection templates from the extracted HOG features from these rendered views. At inference time, a detector is applied to 2D images in a sliding window fashion with multiple scales. When the detector is activated, the pose of 3D shape as well as its 2D location is transferred to the target object. Similar to [20] we also aim for localization and matching of objects in images, but rather than striving to limit the dependence on texture, we prefer to exclude texture completely.

In a recent work from the multimedia retrieval community, Junkert *et al*. [68] limit the dependency on domain shift of the generated textures of 3D shapes by relying on a neural transfer learning scheme rather than HOG descriptors. Their synthesized high quality image renderings of 3D shapes with texture, background and casted shadows. Once these synthesized images are obtained, they extract intermediate feature from an Inception [146] model pre-trained on ImageNet. Given a real image at test time, also represented by the same features, they search for the best matching 3D model by a k nearest neighbor search. As they assume all real and rendered objects have been centralized in the image, they are only able to return viewpoint angles without the location of objects. Similar to [68] we also exploit the transfer abilities

of neural networks. Rather than matching the feature representations of textured 3D model renderings and the query image as is, we prefer to learn the matching function between an image and a texture-free 3D shape.

2.1.2 Contributions

We propose a search engine that given a query object image, localizes and matches the object to the appropriate 3D shape. Different from related approaches, we directly operate on texture-free 3D shapes, enabled by a novel learned matching function. The learned function is a shallow convolutional neural network, merged from a deep two-stream network, encapsulating the relative differences between texture-free 3D shape views and RGB objects. As we are interested in understanding the possibilities as well as the limitations of searching and matching texture-free 3D shapes in images, we rely on a controlled experimental setup on a large and diverse set of texture-free vehicle categories and sub-categories from PASCAL3D+ [170], where we evaluate its sensitivity w.r.t. available 3D shapes and object localization accuracy.

2.2 Towards a 3D-to-2D Search Engine

We cast the problem of searching and matching a 3D shape to a 2D object in an image as a supervised optimization problem. In the offline phase, we assume we have a library of 3D shapes $g^{train} = \{g_1, ..., g_K\}$ describing a variety of object categories and their fine-grained sub-categories, where K is the total number of fine-grained categories. At query time, we start from a single RGB image containing an object of interest, that is either provided or detected automatically. We denote the appearance of the object with x, and in practice it can be the feature activations from one of the layers of a deep convolutional neural network. For the object of interest we assume there is an optimal, albeit hypothetical texture-free 3D model g^*_x. Given a previously unseen image x at query time, our goal is to search among the set of possible 3D shapes and their poses, g^{test}, ϕ^{test}, and place the optimal (g^*_x, ϕ^*_x) on top. To match an RGB image with a texture-free rendering, we introduce a geometric compatibility function $G(\cdot)$, which we want to maximize:

$$\phi_x = \arg\max_{\phi} G(\phi; x, x_{CAD}, g^{test}_k, \phi^{test}_{CAD}) \ , \tag{2.1}$$

where x_{CAD} is the rendered image given a texture-free 3D shape g^{test}_k and pose transformation ϕ_{CAD}. The $g^{test}_k, \phi^{test}_k$ can be equivalent to the shapes observed at training, or expanded to contain more 3D shapes or their poses, $g^{train}_k \subseteq g^{test}_k, \phi^{train}_k \subseteq \phi^{test}_k$. We summarize the data flow of our search engine in Fig. A.1 and detail its main components next.

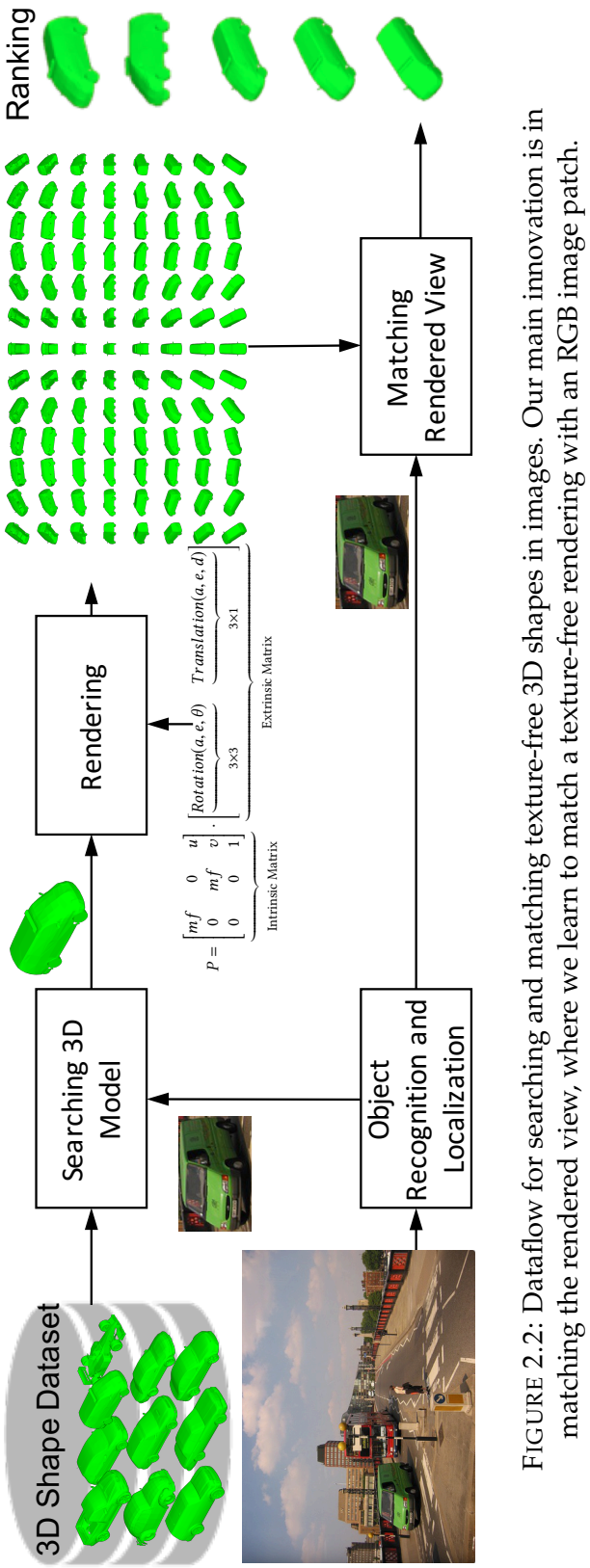


FIGURE 2.2: Dataflow for searching and matching texture-free 3D shapes in images. Our main innovation is in matching the rendered view, where we learn to match a texture-free rendering with an RGB image patch.

2.2.1 Searching and Matching 3D Shapes

Object Recognition and Localization In the literature on 3D to 2D matching, the recognition and localization of the object is often provided in the form of ground truth, *e.g.* [20]. The rationale being that good object detectors are available [131, 101, 130]. Relevant to object recognition and localization, especially for 3D shape matching, are also the so-called amodal bounding boxes [114, 72]. Amodal boxes aim at detecting the full extent of the object, even if the bounds of the surrounding box extent the boundaries of the image. Amodal boxes capture the object centre location (u, v) and corresponding scale d, also including parts that might be not visible because of occlusion or truncation. In our experiments we quantify the sensitivity of our approach w.r.t. the quality of the (amodal) object recognition and localization using both (perturbed) groundtruth as well as a automatic object detection.

Searching 3D Model Once we have obtained the (fine-grained) object category and its enclosing box, we simply query the database of texture-free 3D shapes and forward the selected 3D shape to the rendering stage.

Rendering Each of the 3D shapes g_k can undergo various viewpoint transformations and 3D-to-2D projection. The viewpoint transformation includes rotation and translation that is controlled by extrinsic parameters, *i.e.* azimuth a, elevation e, in-plane rotation θ, camera distance d, whereas intrinsic parameters, *i.e.* principle offset (u, v), focal length f and viewport m, define the camera intrinsic matrix for 3D-to-2D projection. We aggregate the pose transformation parameters to $\phi_{CAD}^{train} = \{\phi_{CAD}^{1}, ..., \phi_{CAD}^{M}\}$, where M is the number of poses seen during training. In the following, to reduce notation clutter we drop the superscript *"train"* whenever it can be derived from the context.

Following [56], the formulation of the projection matrix takes the following form:

$$P = \underbrace{\begin{bmatrix} mf & 0 & u \\ 0 & mf & v \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} \underbrace{Rotation(a,e,\theta)}_{3\times3} & \underbrace{Translation(a,e,d)}_{3\times1} \end{bmatrix}}_{\text{Extrinsic Matrix}}, \tag{2.2}$$

where we assume a calibrated camera, namely the focal length and viewport are fixed to $f = 1, m = 2,000$ respectively [170]. Each set of ϕ_{CAD} values produces a new *3D model rendering*. Similarly, for the hypothetical 3D model g_x^* there exists an optimal transformation vector ϕ_x^*.

Matching Rendered View Ideally, we want to avoid learning an explicit mapping from the appearance modality to the geometric modality. The reason is that an explicit mapping would function well for the 3D model rendering that are observed during training but not generalize well for new 3D

models. To this end we define the geometric compatibility function as a differential on the geometric modality.

During training we have two images, x and x_{CAD}, as well as two rotation matrices, R_x and R_{CAD}, one from the real and one from the rendered image. We define our geometric compatibility function as:

$$G^*_{rel} \propto \partial\phi = g(R_x - R_{CAD}) \ . \tag{2.3}$$

According to eq. (2.3) $\partial\phi \propto g(R_x - R_{CAD})$ is a function of difference between two geometries expressed by R_x and R_{CAD}. Unfortunately, at testing time we cannot have R_x, as this is what we are looking for. To this end we propose to approximate the geometric compatibility with an approximate geometric compatibility function, which receives as inputs the appearance features of the RGB and the CAD rendering, namely:

$$G^*_{rel} \approx G_{rel} \propto \partial x = f(x - x_{CAD}) \ , \tag{2.4}$$

In the spirit of [78], we define the approximate geometric compatibility function $f(\cdot)$ in eq. (2.4) to be a separate neural network module, which we coin *geometric differential module*. A geometric differential module is implemented as a shallow convolutional neural network composed of two layers. The first layer is convolutional with kernel size 1×1 and is fusing the activations from the two streams. The second layer is a fully connected layer with a single output, which approximates $\partial\phi$. Unlike the typical fully connected layer which operates as a function approximator, the geometric differential module approximates a function differential, see Fig. 2.3.

The geometric differential module does not output directly any rotation matrices R_x nor any 3D model rendering ϕ_x. Instead, it returns relative geometric compatibility values. They are relative, because they are supposed to capture the difference of the R_{CAD} from R_x, and by extension the difference of ϕ_{CAD} from ϕ_x, solely on the basis of the appearance features x and x_{CAD}. Effectively, the network learns to estimate directly the matching between an RGB image and any 3D shape rendering, avoiding to compute the object's rotation matrix first. This has two advantages.

First, the network does not directly associate the compatibility score function in eq. (2.4) with the appearance of the particular renderings at training time. In fact, the network does not even make any strict assumptions regarding either the texture appearance of the 3D shape rendering, or the geometric precision of the 3D model for the given RGB object. Thus, even if the 3D model is not a perfect fit to the RGB object, either because the set of viewpoint was limited, or because the geometry of the 3D model is not exactly right for the object of interest, as in Fig. 2.1, the network can still predict the 3D model parameters ϕ_x that match the object of interest. For instance, assume our 3D shape library has models only of a *Boeing 707* and a *Boeing*

717, while our image depicts a *Boeing 747*. Obviously, none of the two available shapes are perfect for our image. Nonetheless, our network returns the best possible fit, as the matching maximizes the matching similarity with the available models, instead of directly classifying the object appearance [3, 20, 96].

Second, since the inference returns only a compatibility score for each provided 3D model rendering, there is no limit to the type of 3D models permissible at test time. The proposed network can return a compatibility score even for 3D models that were not observed during training. As expected, the accuracy of the compatibility score in these cases might be lower, since the network matches 3D shapes and RGB images of object categories it has never seen. Still, the approximate geometric compatibility function has learned *how to match at inference time* on the basis of any 3D model rendering provided at test time. Hence, in theory, the proposed network can cope with 3D shape rendering expanded dynamically, either by adding new 3D shapes or considering a finer discretization of the viewpoints.

An important choice for the geometric differential module is what distance measure it learns to imitate. Although any geometric distance can be used, in this work we opt for approximating the geodesic distance between two rotation matrices of the real object and the rendered image respectively, R, R_{CAD}. Namely, eq. (2.3) becomes

$$G^*_{rel} = \frac{\| \log(R_x^T R_{CAD}) \|_F^2}{\sqrt{2}} \ . \tag{2.5}$$

Thereafter, a Euclidean loss is used to measure how accurately the geodesic differential module predicts G_{rel} by relying only on the appearance features x, x_{CAD}.

2.2.2 Learning to Match

We implement the matching of rendered views as a two-stream architecture [142] with non-shared weights, where each stream is a convolutional network. The first stream receives as an input the cropped RGB images of the object of interest. The second stream receives as an input a cropped rendering from a particular texture-free 3D shape and essentially, describes the object's candidate geometry.

For both streams we adopt the convolutional layers 1 to 5 from AlexNet [83]. The geometric differential module is composed of a convolutional pooling layer [38] with kernel size 1×1, that fuses the two streams, followed by a fully connected module with a single output, predicting the geometric compatibility score between the RGB and the rendering streams. We initialize the weights for convolutional layers 1-5 from AlexNet, while we initialize

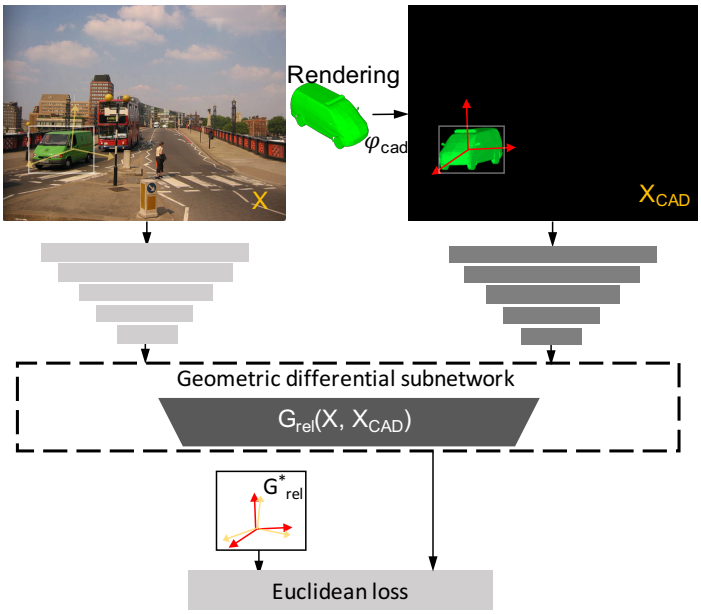


FIGURE 2.3: **Matching a 3D model rendering to a 2D object in an image with geometric differentials.** Starting from an image depicting a fine-grained object category and a texture-free 3D shapes, we derive a 3D model rendering and learn to match them to the localized object of interest. The network is composed of two streams. The first one processes the RGB input. The second one processes the rendered image produced by a candidate 3D model instantiation. The feature maps of the two inputs are fused together and then the geometric differential module estimates the geodesic distance between the two inputs based only on the appearances. It results in a ranking of the 3D model renderings for the localized object, ideally matching the pose of the object.

the remaining layers with a Gaussian distribution with standard deviation 0.005. The network is trained with SGD for 70000 iterations at a learning rate of 0.0001.

Data Preparation. The first stream receives a real image as input, while the second stream a rendered image from a 3D shape. We first create a rendering canvas with the same size as the real image. Given the selected 3D model and viewpoint annotation, we render it onto canvas. This results in a rendered image and a projected bounding box for the 3D model. We crop the real image and rendered image with the projected bounding box to obtain the input data for 2-stream network (see Fig. 2.3).

Training. During training we first sample a real image, for which we have the ground truth 3D model instantiation. As we want to learn to estimate the

geodesic distance between different rotation matrices, we must give good estimates for both when the objects are close as well as far away, geometrically speaking. As our network is a regression model, there exist no positive or negative samples. However, to make sure that our regression model learns to approximate accurately enough across the whole spectrum of geodesic distances, we opt for a stratified sampling of the space of the rotation matrices to collect training examples. Specifically, per training image we sample (*i*) rotation matrices that are almost equal to the ground truth one, (*ii*) rotation matrices that are close enough but not equal, (*iii*) as well as rotation matrices that are far from the ground truth rotation matrix. In total, we end up with 30 rotation matrices per object of interest in training images. We then proceed with the training as usual, relying on SGD for backpropagation.

Inference. During inference and given one image, we traverse over possible renderings. Namely, we render all available 3D shapes in all desired viewpoints. We retain the most confident 3D model instantiation, namely, the 3D shape and viewpoint, whose rendering generates the smallest geodesic distance to the input RGB image.

2.3 Experimental Setup

2.3.1 Texture-free 3D Shape Dataset

We evaluate our approach on the PASCAL3D+ dataset [170]. PASCAL3D+ extends the PASCAL VOC 2012 [36] by matching to every object location a corresponding texture-free 3D shape in its specific pose. As the 3D shapes are category- and not instance-specific and the size of the 3D shape library is finite, the matching quality varies across categories. We perform our experiments on the three vehicle categories with the most examples, namely *aeroplane*, *boat*, and *car*, for which 24 texture-free shapes exist in total, see Fig. 2.6. While this dataset is typically used for the problem of viewpoint estimation [145, 151], we rely on the texture-free 3D shapes for searching and matching. We follow the provided train/test splits. The statistics on the image/object data used in our experiments are summarized in Table 2.1.

2.3.2 Experiments

Experiment 1: Search and match the best rendered view given a specific 3D shape. In our first experiment, we assume the object is perfectly localized in the 2D image and the sub-category of the corresponding 3D shape is known. Thus we simply need to match the object of interest in the real image to a set of rendered views of the corresponding shape. Naturally this is an idealized setting, nonetheless it provides us with the opportunity to establish

an upper-bound for our matching network to compare against in follow-up experiments.

Experiment 2: Search and match the best rendered view among a collection of 3D shapes. Although fine-grained object recognition approaches, *e.g.* [98], can be used for 3D shape selection directly, the matching module of our system should also be able to do this. In this experiment we study the performance of our system doing 3D shape search and rendered view matching at the same time. To be more specific, given a test object of interest in a 2D image, we only know the super category it belongs to (*e.g.* aeroplane), but without knowledge of whether it is an airliner or jet fighter. Hence, for each test object of interest, we have to match it with respect to all possible rendered views and all available fine-grained sub-category 3D shapes. This potentially add difficulties to our approach in distinguishing the best match especially when a few 3D shape candidates are of similar shape (*e.g.* car01, car02 and car05 in Fig. 2.6).

Experiment 3: Search and match the best rendered view from unseen 3D shapes Despite the discrepancy amongst 3D shapes under each category in experiment 1 and 2, all of them are seen both at training and test time. Thus, one interesting question is whether our approach is able to find and match unseen 3D shapes within one super category. Specifically, we consider using only half of the three super categories, aeroplane, boat and car, for training. At test time, we apply the trained network on an unseen sub-category to study whether our network is able to transfer the knowledge from the set of seen sub-categories. Note that, as an ablation study, we continue the setting from experiment 1, where location and scale of the 2D object is known.

Experiment 4: Search and match the best rendered view when object location and scale are imperfect. In this experiment, we investigate how much localization noise our system can tolerate. We first study imperfect object location and scale separately. To simulate the object location error, we perturb the ground truth annotation (u, v) by adding random noise $(\Delta u, \Delta v)$ proportional to the size of ground truth amodal bounding box (width, height). We further evaluate inaccurate detection of object scale in terms of the camera distance d, by randomly scaling it down/up to a maximum of {80%, 90%, 100%, 110%, 120%} of the original value. This results in the 3D shape being rendered smaller/bigger than the object in the real image. Note that scaling down camera distance d means bringing an object closer to the camera and thus a larger amodal bounding box on the 2D space. In the third sub-experiment, we perturb (u, v) and d at the same time, keeping the noise of $(\Delta u, \Delta v)$ at a level of 10% , while randomly varying the scale d to {90%, 100%, 110%} of the original. Note that the fine-grained shape is given in this experiment for a better understanding of the effect of noise on our system.

Experiment 5: Search engine comparison. In our last experiment, we

compare our system with Choy *et al*. [20]. The comparisons are conducted with two settings: (1) search and match given the ground truth object location and (2) fully automatic search and match. As an object detector returns the bounding box coordinates, but not the full object extent in terms of principle point u, v and distance d, we simply choose the center of the bounding box provided by [101] and a scale value d that best matches the bounding box. In contrast, [20] relies on the detected bounding box and a local search is conducted with a multi-scaled detector template. For a fair comparison with [20], we run their software on our rendered images from the 24 texture-free 3D shapes and train both methods in the same way.

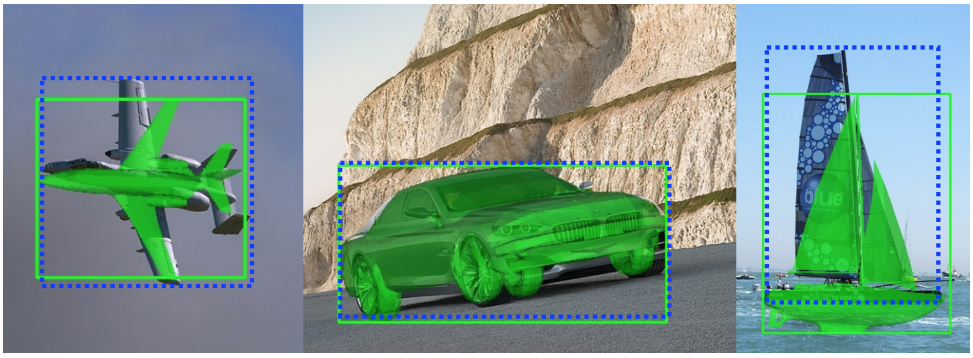


FIGURE 2.4: **2D annotation bounding box (dashed blue) vs amodal bounding box (solid green).** The difference of the two bounding boxes indicates the shape disagreement between a 3D model to the 2D object in image. For the car the two boxes align almost perfectly, while for the sailing boat there is a noticeable discrepancy. In general, in the PASCAL3D+ dataset, the 3D shapes of cars have the best agreement with the 2D object in an image, while boats have the worst. We show statistics of the amodal intersection over union (AIOU) in Table 2.1.

2.3.3 Evaluation Criteria

To test the searching and matching accuracy of our network, we first discretize the rotation parameter space of azimuth a, elevation e and in-plane rotation θ uniformly into 21, 11, 11 bins respectively. This results in 2,541 rendering views in total. At test time we report results when considering the ground truth rendering as well, resulting in 2,542 rendering views and thus 2,542 predictions of the matching score. We report the *precision*@5, namely a prediction is correct if it contains the ground truth 3D shape in the top-5 ranked positions. On top of the alignment precision at 5 we also measure the *amodal intersection over union (AIOU)*, see Fig. 2.4, to quantify the shape disagreement given a 2D object of interest and a 3D shape. To compute the AIOU we measure the intersection over union overlap between the

2D ground truth bounding box and the amodal bounding box[72] derived by the optimal 3D shape from the annotators. A perfectly fitting 3D shape will have very high IOU with the 2D box, while a poor fitting 3D shape will return a very different amodal bounding box and thus low AIOU. We also report the mean over all test queries, indicated by Query Mean.

For the comparison with Choy *et al*. [20], we follow their setup and report accuracy at θ ($ACC@\theta$) when the ground truth bounding box is given. This metric evaluates the fraction of viewpoint predictions that are within a fixed threshold (θ) of its ground truth. When incorporating automatic objection detection, we report Average Viewpoint Precision (AVP) for evaluation. It is similar to Average Precision (AP) in object detection, but it only counts detections as correct when the bounding box overlap ratio exceeds 0.5 and when the prediction of the discretized azimuth angle is in the right bin. We report performance on different levels of azimuth angle discretization, {4,8,16,24} bins. The finer the discretization becomes the harder the task is.

2.4 Results

2.4.1 Search and match specific 3D shape

We present the matching accuracy and AIOU results in Table 2.1. Cars are matched the best, as cars generally have a simple box-like shape. In contrast, boats are more challenging . For one, boats exhibit large variation in shape and object size/scale: a boat includes sub-categories from tanker to canoes, see Fig. 2.6. Moreover, for boats occlusion exists naturally since half of the boat is almost always underwater, thus confusing the matching network that expects the object to be fully visible. When excluding the ground truth pose from the rendered view search space, the *precision*@5 drops from 0.64 to 0.48 for aeroplanes, from 0.44 to 0.27 for boats and from 0.75 to 0.68 for cars. In this scenario, a poor-fitted amodal bounding box hurts even more. To decouple the matching from the rendered view search strategy, we include the ground truth rendering in the rendering search space in the remaining experiments.

When considering individual sub-categories we observe that apart from having a sufficient number of examples available for training, the consistency in shape appearance is important. For example, boat03 is relatively hard. This is due to the fact that sailing boats can have large shape variance and the sail may have different orientation from its body. In contrast, boat04 and boat06 are relatively easy because both of them are big ships resembling a 3D box floating on the water, which facilitates the matching. Similar observations hold for airplanes. Cars generally have more consistent accuracy. Interesting cases are car07 and car04, with car07 being better matched than car04 despite having fewer training samples. The reason is the frontal and

TABLE 2.1: Experiment 1: Search and match the best rendered view given a specific 3D shape. Note the correlation between precision at 5 and quality of the amodal bounding box per category measured by AIOU.

aeroplane					**boat**					**car**				
Subtype	#Train	#Queries	AIOU	p@5	Subtype	#Train	#Queries	AIOU	p@5	Subtype	#Train	#Queries	AIOU	p@5
aeroplane01	761	128	0.80	0.73	boat01	572	123	0.60	0.51	car01	696	97	0.84	0.78
aeroplane02	105	0	0.81	-	boat02	363	24	0.73	0.25	car02	729	37	0.87	0.73
aeroplane03	80	11	0.75	0.36	boat03	519	46	0.68	0.22	car03	932	35	0.87	0.63
aeroplane04	82	26	0.75	0.50	boat04	530	16	0.78	0.62	car04	141	24	0.82	0.54
aeroplane05	88	45	0.76	0.64	boat05	40	11	0.66	0.55	car05	139	17	0.82	0.76
aeroplane06	478	44	0.76	0.50	boat06	492	12	0.69	0.58	car06	1261	18	0.85	0.83
aeroplane07	392	21	0.73	0.62						car07	137	5	0.79	1.00
aeroplane08	88	0	0.80	-						car08	107	8	0.83	0.75
										car09	884	42	0.82	0.98
										car10	641	25	0.86	0.56
Query Mean			0.78	0.64	Query Mean			0.69	0.44	Query Mean			0.85	0.75

TABLE 2.2: **Experiment 2: Search and match the best rendered view among a collection of 3D shapes.** Per super-category, we go over all N 3D shapes per sub-category, namely $N \cdot 2,542$ 3D shape hypotheses. Matching is feasible even when the sub-categories are unknown (compare with Table 2.1).

aeroplane		**boat**		**car**	
Subtype	p@5	Subtype	p@5	Subtype	p@5
aeroplane01	0.35	boat01	0.15	car01	0.46
aeroplane02	-	boat02	0.00	car02	0.27
aeroplane03	0.27	boat03	0.22	car03	0.23
aeroplane04	0.31	boat04	0.44	car04	0.08
aeroplane05	0.24	boat05	0.09	car05	0.35
aeroplane06	0.34	boat06	0.42	car06	0.28
aeroplane07	0.33			car07	0.40
aeroplane08	-			car08	0.25
				car09	0.69
				car10	0.16
Query Mean	0.32	Query Mean	0.18	Query Mean	0.37

rear view of car07 are quite distinguishable, whereas car04 looks like a box, with front and rear poses being often confused.

2.4.2 Search and match among 3D shapes

We show results in Table. 2.2. When the fine-grained sub-category of the object is unknown the network must iterate over all possible 3D shapes. This amounts to an N-fold increase of possible 3D shape instantiations, where N is the number of possible sub-categories. Despite this N-fold increase, we observe that the matching network accuracy drops only 2-fold, approximately. As a reference, a random baseline is about $2 \cdot 10^{-4}$ for aeroplanes, where our matching network scores 0.32. Loosely inspired by the work of Junkert *et al.* [68], intended for textured renderings, we also report a baseline based on a nearest neighbor search strategy. Specifically, we extract fc7 features of a pre-trained AlexNet [83] from the object of interest in both the real and rendered images with $K = 2,542$ different viewpoints, and rank them based on cosine similarity. The results show this approach does not work well in our setting, with p@5 around 0.07, as the domain shift from real images to rendered images without texture and background context is simply too large.

TABLE 2.3: **Experiment 3: Search and match the best rendered view from unseen 3D shapes.** Intra-category knowledge transfer can be performed when new 3D shapes added at test time are somewhat similar to the ones seen during training (marked in gray).

aeroplane		**boat**		**car**	
Subtype	p@5	Subtype	p@5	Subtype	p@5
aeroplane01	0.67	boat01	0.43	car01	0.85
aeroplane02	-	boat02	0.29	car02	0.84
aeroplane03	0.45	boat03	0.37	car03	0.57
aeroplane04	0.62			car04	0.62
				car05	0.71
aeroplane05	0.40	boat04	0.06	car06	0.61
aeroplane06	0.14	boat05	0.55	car07	1.00
aeroplane07	0.33	boat06	0.00	car08	0.25
aeroplane08	-			car09	0.88
				car10	0.44

2.4.3 Search and match unseen 3D shapes

We report the performance of our network trained on both seen and unseen sub-categories in Table 2.3. The network is able to transfer its matching knowledge to unseen shapes to some extent. Note that this experimental setting is quite close to zero-shot image classification [81] where one classifies a category without having seen its visual examples, a challenging task where accuracy drops are generally high. Focusing on sub-categories, aeroplane05 is matched best, while aeroplane06 is affected more. Likely because aeroplane05 looks similar to both aeroplane01 and aeroplane02, while aeroplane06 (jet fighter) is more different. Similarly, boat05 is well matched because it resembles boat02. However, boat04 and boat06 refer to big ships, and transferring the matching knowledge from the considerably smaller boats (yachts, canoes and sailing boats) is harder. For cars, where all sub-categories are quite similar we observe a good matching accuracy. We conclude that if new 3D shapes added at test time are somewhat similar to the seen ones, our network can match them reasonably well.

2.4.4 Search and match under noisy conditions

Noisy object location. We first look at the effect of noise on the object location (u, v), see Fig. 2.5 (A). For a limited amount of 5% noise the impact on the network is modest, when averaged over categories the loss is 0.59 for aeroplane, 0.41 for boat and 0.73 for car. When we add more noise performance starts to suffer, with 10% noise the numbers drop to 0.52, 0.32 and

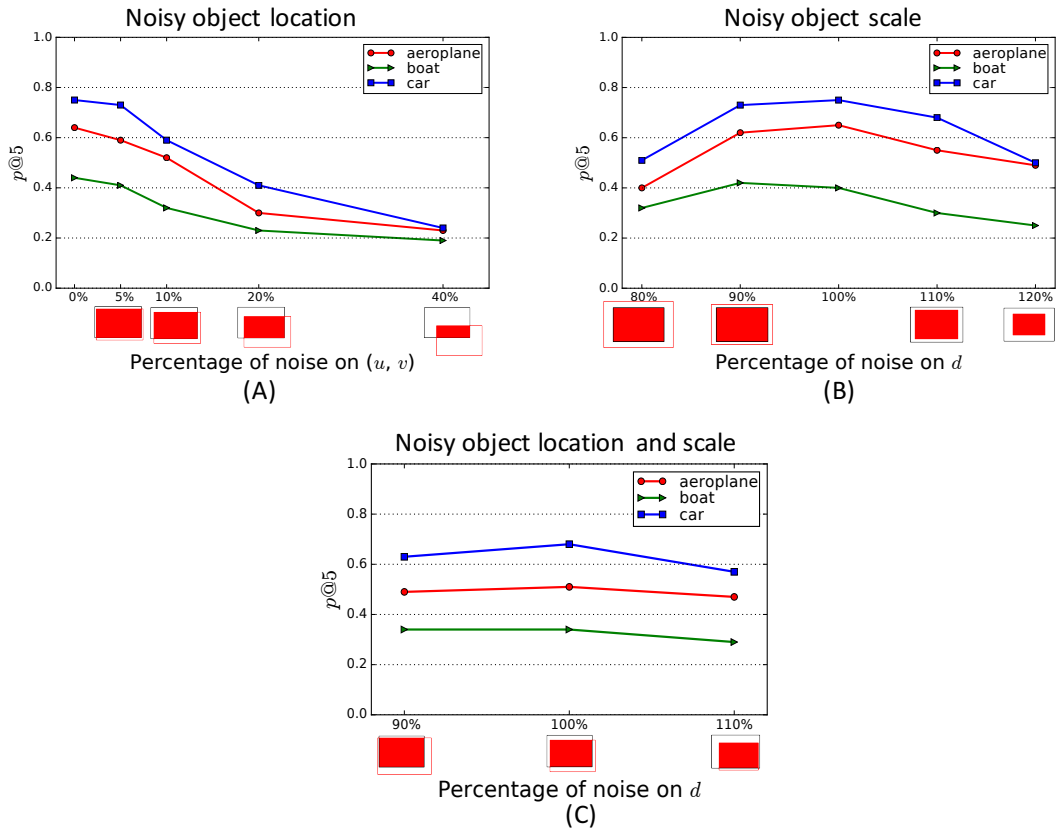


FIGURE 2.5: **Experiment 4: Search and match the best rendered view when object location (u, v) and d scale are imperfect.** On the x axis we visualize the indicative overlap displacement with respect to the perfect box. For moderate noise (5-10%) the model recovers a good match.

0.59 respectively. Unsurprisingly the performance drops as more and more noise is added, be it that the drop is less pronounced after 20%.

Noisy object scale. Next, we investigate the sensitivity when adding noise to the camera distance d, which relates to the detected object scale (smaller $d \rightarrow$ larger object scale), see Fig. 2.5 (B). As before, the larger the deviation the larger the drop in performance and any noise in the range $\pm 10\%$ leads to an acceptable matching accuracy. Interestingly, when scaling d down to 90% it leads to a slight increase in performance for boats. The reason is that scaling down the camera distance d results in a bigger amodal bounding box that often is a better fit to the actual object.

Noisy object location and scale. Last, we assess the impact of having noise in both the location and the scale. Results are shown in Fig. 2.5 (C). We observe the accuracy remains stable for all categories, indicating the two different types of noise do not reinforce each other too much.

TABLE 2.4: Experiment 5: Search engine comparison with Choy *et al*. [20] using ground truth object location, where we run the software of Choy *et al*. on our texture-free setting. In terms of *ACC@θ* our approach is especially beneficial for 3D shapes that have sufficient training samples, where Choy *et al*. profit from limited example regimes (see Table 2.1). For boats both approaches perform modestly.

aeroplane			**boat**			**car**		
Subtype	Choy *et al*.	This chapter	Subtype	Choy *et al*.	This chapter	Subtype	Choy *et al*.	This chapter
aero01	0.48	0.59	boat01	0.32	0.36	car01	0.39	0.62
aero02	-	-	boat02	0.48	0.21	car02	0.62	0.60
aero03	0.55	0.46	boat03	0.33	0.26	car03	0.54	0.57
aero04	0.39	0.31	boat04	0.19	0.19	car04	0.42	0.50
aero05	0.40	0.56	boat05	0.50	0.09	car05	0.35	0.71
aero06	0.39	0.57	boat06	0.42	0.25	car06	0.83	0.61
aero07	0.33	0.52				car07	0.40	0.80
aero08	-	-				car08	0.63	0.50
						car09	0.76	0.74
						car10	0.60	0.48
Query Mean	0.35	0.54	Query Mean	0.22	0.29	Query Mean	0.29	0.61

TABLE 2.5: Experiment 5: Search engine comparison with Choy *et al*. [20] with automatically detected object location, using their public software on our texture-free setting. The proposed system achieves better performance in terms of *AVP* for aeroplane and car and worse for boat.

	aeroplane				**boat**				**car**			
#Bins	4	8	16	24	4	8	16	24	4	8	16	24
Choy *et al*.	0.35	0.22	0.10	0.05	0.14	0.06	0.02	0.01	0.23	0.17	0.10	0.07
This chapter	0.44	0.24	0.10	0.05	0.07	0.04	0.01	0.01	0.30	0.26	0.19	0.11

2.4.5 Search engine comparison

Results for given ground truth object location are shown in Table 2.4. In terms of the mean over all queries, we obtain better *ACC@θ* than [20] for all 3 categories. The boat category has relatively low performance for both methods, because of the low AIOU rate (see Table 2.1) that indicates large shape discrepancies. Looking into the sub-category comparisons, Choy *et al*. is better in some cases (e.g. aeroplane03, aeroplane04, boat02-boat06), mostly when less training examples are provided which results in our two-stream

network being underfitting. Table 2.5 reports results of joint object detection and 3D-to-2D matching. In terms of AVP we outperform Choy *et al*. also in this setting. As expected, performance for both search engines suffers when azimuth angle discretization becomes finer. While aeroplanes and cars are again matched better, now aeroplane are easier to match than cars, presumably due to their easier detection of typically simpler backgrounds. We conclude that our system with a learned matching function is beneficial over hand-engineered feature matching approaches when sufficient (>80) training samples are available. At the same time there is still a lot of work needed before we arrive at generic and precise searching and matching of 3D shapes in images.

2.5 Conclusion

This chapter focuses on searching and matching the best rendered view of a texture-free 3D shape to an object of interest in a 2D image. Matching rendered views of 3D shapes to RGB images is challenging because of imperfect 3D shapes and domain shift in appearance due to texture mismatch. We propose a deeply learned matching function that attacks these challenges and can be used as a search engine of 3D shapes to objects in 2D. We evaluate the proposed search engine on the most populated PASCAL3D+ vehicle categories, testing the capabilities of transferring the learnt function to unseen 3D shapes and its sensitivity to imperfect 3D shapes and localization. We also identify the need for accurate amodal bounding box detection in 2D images as an important 3D-to-2D matching topic for further investigation.

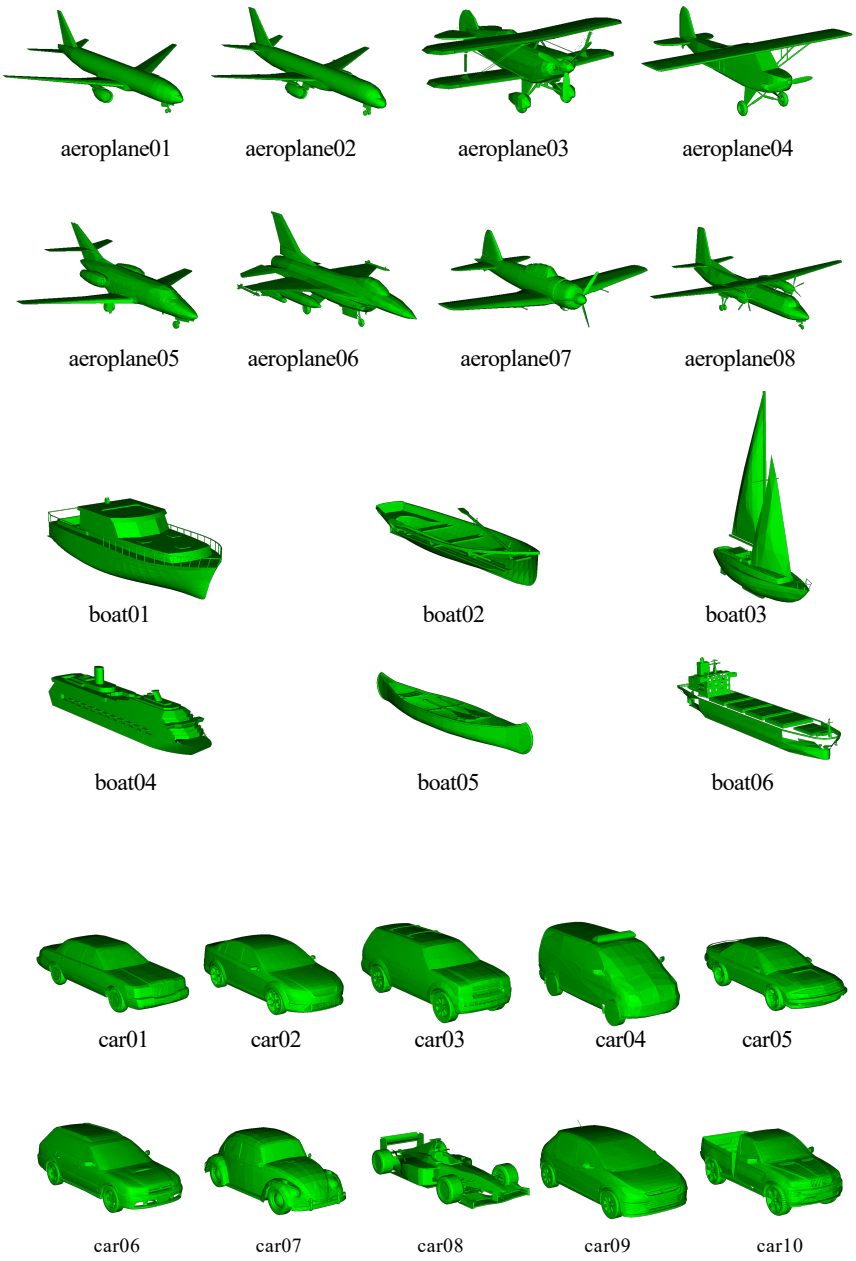


FIGURE 2.6: **Texture-free 3D shapes** for the 24 fine-grained *aeroplane*, *boat*, and *car* objects used in our experiments.

Chapter 3

Spherical Regression

3.1 Introduction

Computer vision challenges requiring continuous outputs are abundant. Viewpoint estimation [151, 145, 124, 125], object tracking [149, 46, 65, 71], and surface normal estimation [4, 35, 179, 126] are just three examples. Despite the continuous nature of these problems, regression based solutions that seem a natural fit are not very popular. Instead, classification based approaches are more reliable in practice and, thus, dominate the literature [151, 145, 110, 149, 85]. This leads us to an interesting paradox: while several challenges are of continuous nature, their present-day solutions tend to be discrete.

In this work we start from this paradox and investigate why regression lags behind. When juxtaposing the mechanics of classification and regression we observe that classification is naturally contained within a probability n-simplex geometry defined by the popular softmax activation function. The gradients propagated backwards to the model are constrained and enable stable training and convergence. In contrast, regression is not contained by any closed geometry. Hence, the gradients propagated backwards are not constrained, potentially leading to unstable training or convergence to suboptimal local minima. Although classification solutions for continuous problems suffer from discretization errors in annotations and predictions, they typically lead to more reliable learning [110, 85].

Founded on the relation between classification, regression and closed geometric manifolds, we revisit regression in deep networks. Specifically, we observe many continuous output problems in computer vision are naturally contained in closed geometrical manifolds defined by the problem at hand. For instance, in viewpoint estimation, angles cannot go beyond the $[-\pi, \pi]$ range. Or, in surface normal estimation the ℓ_2 norm of the surface normals must sum up to 1 to form unit vectors that indicate directionality. It turns out that a natural framework for posing such continuous output problems are the n-spheres S^n [40, 34], which are naturally closed geometric manifolds defined in the $\mathbb{R}^{(n+1)}$ space. We, therefore, rethink regression in continuous spaces in the context of n-spheres, when permitted by the application.

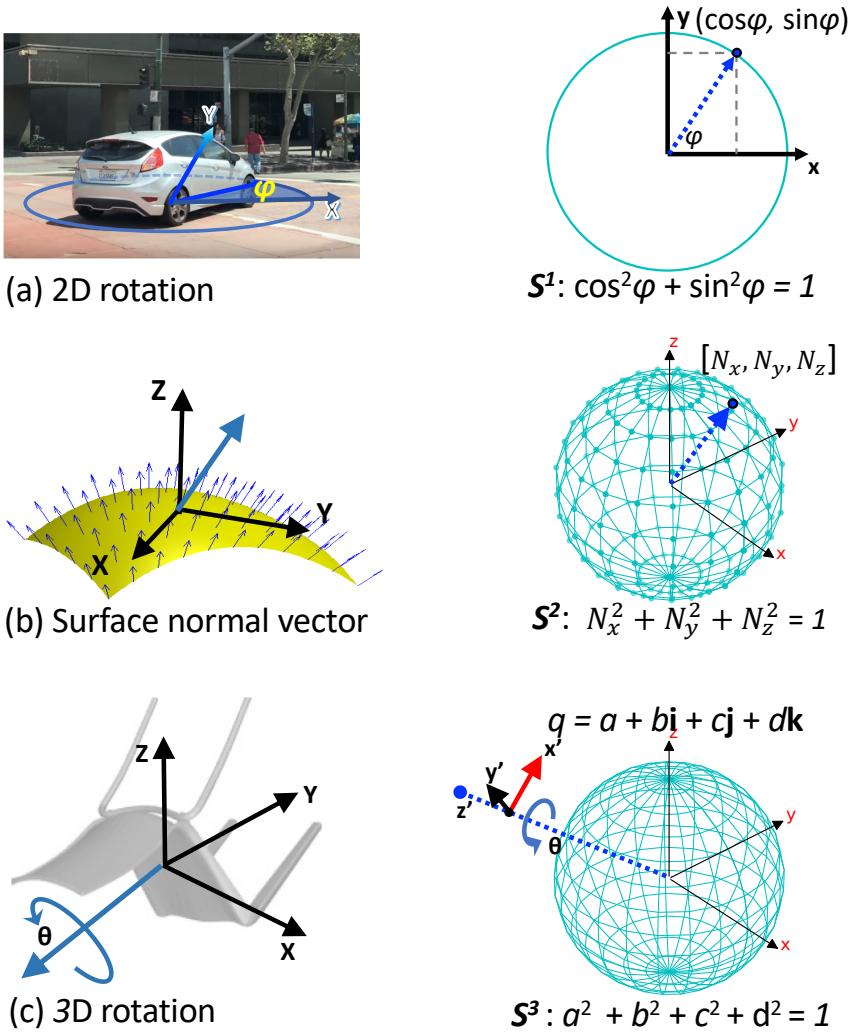


FIGURE 3.1: **Many computer vision problems can be converted into a n-sphere problem.** n-spheres are naturally closed geometric manifolds defined in the $\mathbb{R}^{(n+1)}$ space. Examples are a) viewpoint estimation, b) surface normal estimation, and c) 3D rotation estimation. This chapter proposes a general regression framework that can be applied on all these n-sphere problems.

It turns out that if we introduce a proposed spherical exponential mapping on n-spheres at the regression output we obtain regression gradients that are constrained and well-behaving, similar to classification-based learning. We refer to regression using the proposed spherical exponential mappings on S^n spheres as S^n spherical regression.

In this work we make the following contributions. First, we link the framework of n-spheres to continuous output computer vision tasks. By doing so, they are amenable to the properties of the n-spheres formulation,

leading to spherical regression. Second, we propose a novel nonlinearity, the spherical exponential activation function, specifically designed for regressing on S^n spheres. We show the activation function improves the results obtained by regular regression. Third, we show how the general spherical regression framework can be utilized for particular computer vision challenges. Specifically, we show how to recast existing methods for viewpoint estimation, surface normal estimation and 3D rotation estimation to the proposed spherical regression framework. Our experiments demonstrate the benefit of spherical regression for these problems.

We now first describe in Section 3.2 the motivation behind the deep learning mechanics of classification and regression. Based on the insights derived, we describe in Section 3.3 the general framework for spherical regression on S^n spheres. We then explain how to specialize the general frameworks for particular applications, see Fig. 3.1. We describe the related work for these tasks in Section 5.2. In Section 5.5, we evaluate spherical regression for the three applications.

3.2 Motivation

Deep classification and regression networks. We start from an input image $\mathbf{x}$ of an object with a supervised learning task in mind, be it classification or regression. Regardless the task, if we use a convolutional neural network (CNN) we can split it into two subnetworks, the base network and the prediction head, see (eq. 3.1).

$$\underbrace{\mathbf{x} \xrightarrow[\text{base network}]{H(\cdot)} \boldsymbol{O} = \begin{bmatrix} o_0 \\ o_1 \\ \vdots \\ o_n \end{bmatrix}}_{\text{Base network}} \underbrace{\xrightarrow[\text{activation}]{g(\cdot)} \boldsymbol{P} = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} \xleftrightarrow[\text{loss}]{\mathcal{L}(\cdot,\cdot)} \boldsymbol{Y}}_{\text{Prediction head}} \tag{3.1}$$

The base network considers all the layers from input $\mathbf{x}$ till layer $\boldsymbol{O}$. It defines a function $\boldsymbol{O} = H(\mathbf{x})$ that returns an intermediate latent embedding $\boldsymbol{O} = [o_0, o_1, ..., o_n]^\top$ of the raw input $\mathbf{x}$. The function comprises a cascade of convolutional layers intertwined with nonlinearities and followed by fully connected layers, $H = h_l \circ h_{l-1} \cdots \circ h_k \circ \cdots \circ h_2 \circ h_1$, where h_k is the θ-parameterized mapping of k-th layer. Given an arbitrary input signal $\mathbf{x}$, the latent representation $\boldsymbol{O}$ is unconstrained, namely $\mathbf{x} = H(\mathbf{x}) \rightarrow \mathbb{R}^{(n+1)}$.

The prediction head contains the last $(n+1)$-dimensional layer $\boldsymbol{P}$ before the loss function, which is typically referred to as the network output. The output is obtained from an activation function $g(\cdot)$, which generates the output $\boldsymbol{P}$: $p_k = g(o_k; \boldsymbol{O})$ using as input the intermediate raw embedding $\boldsymbol{O}$

returned by the base network. The activation function $g(\cdot)$ imposes a structure to the raw embdedding $\boldsymbol{O}$ according to the task at hand. For instance, for a CNN trained for image classification out of $1,000$ classes we have a $1,000$-dimensional output layer $\boldsymbol{P}$ that represents softmax probabilities. And, for a CNN trained for 2D viewpoint estimation we have a 2-dimensional output layer $\boldsymbol{P}$ that represents the trigonometric functions $\boldsymbol{P} = [cos\phi, sin\phi]$. After the prediction head lies the loss function $\mathcal{L}(\boldsymbol{P}, \boldsymbol{Y})$ that computes the distance between the prediction $\boldsymbol{P}$ and the ground truth $\boldsymbol{Y} = [y_0, y_1, ...]^\top$, be it cross entropy for classification or sum of squared errors for regression.

The dimensionalities of $\boldsymbol{O}$ and $\boldsymbol{P}$ vary according to the type of classification or regression that is considered. For classification $\boldsymbol{P}$ represents the probability of $(n+1)$ discretized bins. For regression, $\boldsymbol{P}$ depends on the assumed output representation dimensionality, *e.g.*, regression 1D [124], regression 2D [124, 8] or regression 3D [120] and beyond can have different output dimensions. Together the subnetworks comprise a standard deep architecture, which is trained end-to-end.

Training. During training, the k-th layer parameters are updated with stochastic gradient descent, $\theta_k \leftarrow \theta_k - \gamma \frac{\partial \mathcal{L}}{\partial \theta_k}$, where γ is the learning rate. Expanding by the chain rule of calculus we have that

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}} \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{O}} \left(\frac{\partial \boldsymbol{O}}{\partial h_{l-1}} \cdots \frac{\partial h_{k+1}}{\partial h_k} \right) \frac{\partial h_k}{\partial \theta_k} \tag{3.2}$$

Training is stable and leads to consistent convergence when the gradients are constrained, otherwise gradient updates may cause bouncing effects on the optimization landscape and may cancel each other out. Next, we examine the behavior of the output activation $\boldsymbol{P}$ and the loss functions for classification and regression.

Classification. For classification the standard output activation and loss functions are the softmax and the cross entropy, that is $g(o_i; \boldsymbol{O}) = \{p_i = e^{o_i} / \sum_j e^{o_j}, i = 0 \cdots n\}, \mathcal{L}(\boldsymbol{O}, \boldsymbol{Y}) = -\sum_i y_i log(p_i)$. The p_i and y_i are the posterior probability and the one-hot vector for the i-th class, and d is the number of classes. Note that softmax maps the raw latent embedding $O \in \mathbb{R}^{(n+1)}$ to a structured output $\boldsymbol{P}$, known as n-simplex, where each dimension is positive and the sum equals to one, *i.e.* $\sum_i p_i = 1$ and $p_i > 0$. The partial derivative of the probability output with respect to the latent activation equals to

$$\frac{\partial p_j}{\partial o_i} = \begin{cases} p_j \cdot (1 - p_j), & \text{when } j = i \\ -p_i \cdot p_j, & \text{when } j \neq i \end{cases} \tag{3.3}$$

Crucially, we observe that the partial derivative $\frac{\partial p_j}{\partial o_i}$ does not directly depend on $\boldsymbol{O}$. This leads the partial derivative of the loss function with respect to o_i,

namely

$$\frac{\partial \mathcal{L}}{\partial o_i} = -\sum_k \frac{y_k}{p_k} \cdot \frac{\partial p_k}{\partial o_i} = p_i - y_i, \tag{3.4}$$

to be independent of $\boldsymbol{O}$ itself. As $\boldsymbol{P}$ corresponds to a probability distribution that lies inside the n-dimensional simplex, it is naturally constrained by its ℓ_1 norm, $p_j < 1$. Thus, the partial derivative $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ depends only on a quantity that is already constrained.

Regression. In regression usually there is no explicit activation function in the final layer to enforce some manifold structure. Instead, the raw latent embedding $\boldsymbol{O}$ is directly compared with the ground truth. Take the smooth-L1 loss as an example,

$$\mathcal{L} = \begin{cases} 0.5|y_i - o_i|^2 & \text{if} |y_i - o_i| \leqslant 1 \\ |y_i - o_i| - 0.5 & \text{otherwise.} \end{cases} \tag{3.5}$$

The partial derivative of the loss with respect to o_i equals to

$$\frac{\partial \mathcal{L}}{\partial o_i} = \begin{cases} -(y_i - o_i) & \text{if} |y_i - o_i| \leqslant 1 \\ -\text{sign}(y_i - o_i) & \text{otherwise.} \end{cases} \tag{3.6}$$

Unlike classification, where the partial derivatives are constrained, for regression we observe that the $\frac{\partial \mathcal{L}}{\partial o_i}$ directly depends on the raw output $\boldsymbol{O}$. Hence, if $\boldsymbol{O}$ has high variance, the unconstrained gradient will have a high variance as well. Because of the unconstrained gradients training may be unstable.
Conclusion. Classification with neural networks leads to stable training and convergence. The reason is that the partial derivatives $\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}} \cdot \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{O}}$ are constrained, and, therefore, the gradient updates $\frac{\partial \mathcal{L}}{\partial \theta_k}$, are constrained. The gradients are constrained because the output $\boldsymbol{P}$ itself is constrained by the ℓ_1 norm of the n-simplex, $\sum_i p_i = 1$. Regression with neural networks may have instabilities and sub-optimal results during training because gradient updates are unconstrained. We examine next how we can define a similar closed geometrical manifold also for regression. Specifically, we focus on regression problems where the target label Y lives in a constrained n-sphere manifold.

3.3 Spherical regression

The n-sphere, denoted with S^n, is the surface boundary of an $(n+1)$-dimensional ball in the Euclidean space. Mathematically, the n-sphere is defined as $S^n = \left\{\mathbf{x} \in \mathbb{R}^{n+1} : \|\mathbf{x}\| = r\right\}$ and is constrained by the ℓ_2 norm, namely $\sum_i x_i^2 = 1$. Fig. 3.1 gives examples of simple n-spheres, where S^1 is the circle and S^2 the

surface of a 3D ball. Where the n-simplex constrains classification by the ℓ_1 simplex norm, we next present how to constrain regression by the ℓ_2 norm of an n-sphere.

3.3.1 Constraining regression with n-spheres

To encourage stability in training regression neural networks on S^n spheres, one reasonable objective is to ensure the gradients are constrained. To constrain the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$, we propose to insert an additional activation function in regression after the raw embedding layer $\boldsymbol{O}$. The activation function should have the following properties.

(I) The *output* of the activation, $\boldsymbol{P} = \{p_k\}$, must live on n-sphere, namely its ℓ_2 norm $\sum_{k=1} p_k^2 = 1$ must be constant, *e.g.*, $cos^2\phi + sin^2\phi = 1$. This is necessary for spherical targets.

(II) Similar to classification, the *gradient* $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ must not directly depend on the input signal. That is, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ must not depend directly on the raw latent embedding $\boldsymbol{O} \in \mathbb{R}^{(n+1)}$.

To satisfy property (I), we pick our activation function such that it produces normalized values. We opt for the ℓ_2 normalization form: $p_j = g(o_j; \boldsymbol{O}) = \frac{f(o_j)}{\sqrt{\sum_k f(o_k)^2}}$, where $f(\cdot)$ corresponds to any univariate mapping. The partial derivative of the output with respect to the latent O then becomes:

$$\begin{aligned} \frac{\partial p_j}{\partial o_i} &= \frac{\partial \left[\frac{f(o_j)}{\sqrt{\sum_k f(o_k)^2}} \right]}{\partial o_i} \\ &= \begin{cases} \left(\frac{df(o_i)}{do_i} \cdot \frac{1}{A} \right) \cdot (1 - {p_i}^2), & \text{when } j = i \\ \left(\frac{df(o_i)}{do_i} \cdot \frac{1}{A} \right) \cdot (-p_i \cdot p_j), & \text{when } j \neq i \end{cases} \end{aligned} \tag{3.7}$$

where $A = \sqrt{\sum_k f(o_k)^2}$ is the normalization factor.

Still, $\frac{\partial p_j}{\partial o_i}$ is potentially depending on the raw latent embedding $\boldsymbol{O}$ through the partial function derivatives $\frac{df(o_j)}{do_i}$ and the normalization factor A. To satisfy property (II) and make $\frac{\partial p_i}{\partial o_j}$ independent from the raw output $\boldsymbol{O}$, and thus constrained, we must make sure that $\left(\frac{df(o_j)}{do_j} \cdot \frac{1}{A} \right)$ becomes independent of $\boldsymbol{O}$. In practice, there are a limited number of choices for $f(\cdot)$ to satisfy this constraint. Inspired by the softmax activation function, we resort to the exponential map $f(o_i) = e^{o_i}$, where $\frac{df(o_i)}{do_i} = f(o_i)$ and $\frac{\partial f(o_i)}{\partial o_i} \cdot \frac{1}{A} = \frac{f(o_i)}{A} = p_i$.

Thus Eq. 3.7 is simplified as

$$\frac{\partial p_j}{\partial o_i} = \begin{cases} p_i \cdot (1 - p_i^2), & \text{when } j = i \\ -p_i^2 \cdot p_j, & \text{when } j \neq i \end{cases} \tag{3.8}$$

removing all dependency on $\boldsymbol{O}$.

Since our activation function has a similar form as softmax, which is also known as normalized exponential function, we refer to our activation function as *Spherical Exponential Function*. It maps inputs from $\mathbb{R}^{n+1}$ to the positive domain of the n-Sphere, *i.e.* $S_{exp}(\cdot) : \mathbb{R}^{n+1} \rightarrow \mathbb{S}^n_+$:

$$p_j = S_{exp}(o_j; \boldsymbol{O}) = \frac{e^{o_j}}{\sqrt{\sum_k (e^{o_k})^2}} \tag{3.9}$$

Converting Eq. 3.8 into matrix provides Jacobian as $\mathbf{J}_{S_{exp}} = (\boldsymbol{I} - \boldsymbol{P} \otimes \boldsymbol{P}) \cdot diag(\boldsymbol{P})$ where $\otimes$ denotes outer product (see supplementary material for details). Notice that if we only do ℓ_2 normalization without exponential, the Jacobian is given as $\mathbf{J}_{\mathcal{S}_{flat}} = (\boldsymbol{I} - \boldsymbol{P} \otimes \boldsymbol{P}) \cdot \frac{1}{||\boldsymbol{O}||}$, which is influenced by the magnitude of $\boldsymbol{O}$ in gradient, which is unconstrained.

Unfortunately, the exponential map in $S_{exp}(\cdot)$ restricts the output to be in the positive range only, whereas our target can be either positive or negative. To enable regression on the full range on n-sphere coordinates we rewrite each dimension into two parts: $p_i = sign(p_i) \cdot |p_i|$. We then use the output from the spherical exponential function to learn the absolute values $|p_i|, i = 0, 1, ..., n$ only. At the same time, we rely on a separate classification branch to predict the sign values, $sign(p_i), i = 1, ..., d$ of the output. The overall network is shown in Fig. 3.2:

Conclusion. Given the spherical exponential mapping for $g(\cdot)$, the gradient $\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{O}}$ is detached from $\boldsymbol{O}$, and $\boldsymbol{P}$ is constrained by the n-Sphere. Thus, to make the parameter gradients also constrained, we just need to pick a suitable loss function. It turns out that there are no significant constraints for the loss function. Given ground truth $\boldsymbol{Y}$, we can set the loss to be the negative dot product $\mathcal{L} = -\langle |\boldsymbol{P}|, |\boldsymbol{Y}| \rangle$. Since both $\boldsymbol{P}$ and $\boldsymbol{Y}$ are on sphere with ℓ_2 norm equal to 1 (*i.e.* $||\boldsymbol{P}||_2 = ||\boldsymbol{Y}||_2 = 1$), this is equivalent to optimize with cosine proximity loss or L2 loss *. In this case, the gradients are $\frac{\partial \mathcal{L}}{\partial p_i} = -sign(p_i)|y_i|$ and only relate to $\boldsymbol{P}$. We could also treat the individual outputs $\{p_1^2, p_2^2 ...\}$ as probabilities with a cross-entropy loss on continuous labels y_i^2, in which case we would have that $H(\boldsymbol{Y}^2, \boldsymbol{P}^2) = \sum_i y_i^2 \log \frac{1}{p_i^2}$. We conclude that the Spherical Regression using the spherical exponential mapping allows for constrained

*For cosine proximity loss: $\mathcal{L} = -\frac{\langle |\boldsymbol{P}|, |\boldsymbol{Y}| \rangle}{||\boldsymbol{P}||_2 \cdot ||\boldsymbol{Y}||_2} = -\langle |\boldsymbol{P}|, |\boldsymbol{Y}| \rangle$. For L2 loss: $\mathcal{L} = ||\boldsymbol{P} - \boldsymbol{Y}||_2^2 = ||\boldsymbol{P}||_2^2 + ||\boldsymbol{Y}||_2^2 - 2\langle |\boldsymbol{P}|, |\boldsymbol{Y}| \rangle = 2 - 2\langle |\boldsymbol{P}|, |\boldsymbol{Y}| \rangle$.

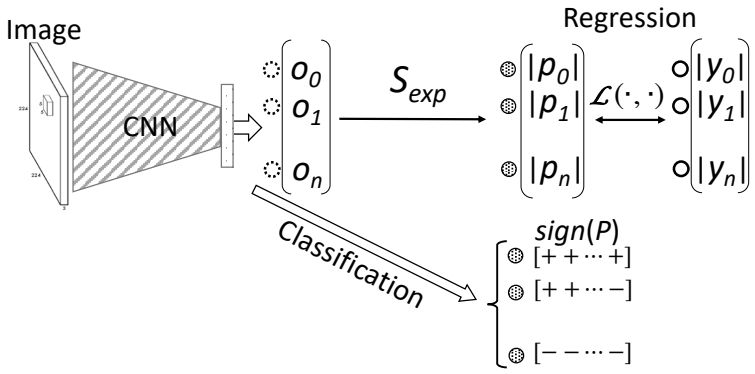


FIGURE 3.2: **Regressing on n-spheres** with targets $Y = [y_0, ..., y_n]$, *i.e.* $\sum_i y_i^2 = 1$. The model processes the input image and first returns a raw latent embedding $\boldsymbol{O} = [o_0, ..., o_n] \in \mathbb{R}^{(n+1)}$. Then, a regression branch using the proposed spherical exponential acctivation S_{exp} maps $\boldsymbol{O}$ to a structured output $|\boldsymbol{P}| = [|p_0|, ..., |p_n|]$. A classification branch is also used to learn the sign labels of $\boldsymbol{P}$. Prediction is made by $\boldsymbol{P} = sign(\boldsymbol{P}) \cdot |\boldsymbol{P}|$.

parameter updates and, thus, we expect it to lead to stable training and convergence. We verify this experimentally on three different applications and datasets.

3.3.2 Specializing to S^1, S^2 and S^3

Next, we show how to specialize the general n-sphere formulation for different regression applications that reside on specific n-spheres.

S^1 case: Euler angles estimation. Euler angles are used to describe the orientation of a rigid body with respect to a fixed coordinate system. They are defined by 3 angles, describing 3 consecutive rotations around fixed axes. Specifically, each of the angles $\phi \in [0, 2\pi]$ can be represented by a point on a unit circle with 2D coordinate $[cos\phi, sin\phi]$, see Fig. 3.1. Since $cos^2\phi + sin^2\phi = 1$, estimating these coordinates is an S^1 sphere problem. Consequently, our prediction head has two components: *i)* a regression branch with spherical exponential activations for absolute values $|\boldsymbol{P}| = [|cos\phi|, |sin\phi|]$ and, *ii)* a classification branch to learn all possible sign combinations between $sign(cos\phi)$ and $sign(sin\phi)$, that is a 4-class classification problem: $sign(\boldsymbol{P}) \in \{(+,+), (+,-), (-,+), (-,-)\}$. We could also predict the signs independently and have fewer possible outputs, however, this would deprive the classifier from the opportunity to learn possible correlations.

During training time, we jointly minimize the regression loss (cosine proximity) and the sign classification loss (cross-entropy). For the inference, we

do the final prediction by merging the absolute values and sign labels together:

$$\begin{cases} cos\phi = sign(cos\phi) \cdot |cos\phi| \\ sin\phi = sign(sin\phi) \cdot |sin\phi| \end{cases} \tag{3.10}$$

Beyond Euler angles, other 2D rotations can be learned in the same fashion.

S^2 case: Surface normal estimation. A surface normal is the direction that is perpendicular to the tangent plane of the point on the surface of objects in a 3D scene, see Fig. 3.1.(b). It can be represented by a unit 3D vector $\mathbf{v} = [N_x, N_y, N_z]$ for which $N_x^2 + N_y^2 + N_z^2 = 1$. Thus, a surface normal lies on the surface of a unit 3D ball, *i.e.* an S^2 sphere. Surface normal estimation from RGB images makes pixel-wise predictions of surface normals of the input scene.

It is worth noticing that all surface normals computed by a 2D image should always be pointing outwards from the image plane, that is $N_z < 0$, since only these surfaces are visible to the camera. This halves the prediction space to a semi-sphere of S^2. Again, when designing the spherical regressor for surface normals, we have a regression branch to learn the absolute normal values $[|N_x|, |N_y|, |N_z|]$ and a classification branch for learning all combinations of signs for N_x and N_y. The total number of possible sign classes is 4, similar to Euler angle estimation. The training and inference is similar to Euler angles as well. Other S^2 problems include learning the direction of motion in 2D/3D flow fields, geographical locations on the Earth sphere and so on.

S^3 case: 3D rotation estimation. Rotational transformations are relevant in many computer vision tasks, for example, orientation estimation, generalized viewpoint and pose estimation beyond Euler angles or camera relocation. Rotational transformations can be expressed as orthogonal matrices of size n with determinant +1 (rotation matrices). We can think of the set of all possible rotation matrices to form a group that acts as an operator on vectors. This group is better known as the *special orthogonal Lie group* $\mathcal{SO}(n)$ [53]. Specifically, the $\mathcal{SO}(2)$ represents the set of all 2D rotation transformations, whereas $\mathcal{SO}(3)$ represents the set of all possible 3D rotations.

We have already shown that 2D rotations can be mapped to a regression on an S^1 sphere, thus the set $\mathcal{SO}(2)$ of all 2D rotations is topologically equivalent to the S^1 sphere. Interestingly, the topology of 3D rotations is not as straightforward [53], namely there is no n-sphere that is equivalent to $\mathcal{SO}(3)$. Instead, as shown in Fig. 3.1.(c) a 3D rotation $\mathcal{SO}(3)$ can be thought of as first choosing a rotation axis $\mathbf{v}$ and then rotating by an angle θ. This approach leads to the well known S^3 representation of quaternions [55], which is the closest equivalent to the 3D rotation [139].

A unit quaternion is equal to $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where $a^2 + b^2 + c^2 + d^2 = 1$. As q and $-q$ give the same rotation, we restrict ourselves to $a > 0$, which again halves the output space. We, therefore, need to predict the signs

of only 3 imaginary components $\{b, c, d\}$ to a total of 8 (2^3) classes. The design of the prediction heads and the loss functions are similar to the case of surface normal prediction on S^2, only now having 8 sign classes. Given the axis-angle representation $(\theta, \mathbf{v})$ of $\mathcal{SO}(3)$, we can, therefore, rewrite a quaternion into $q = (cos\frac{\theta}{2}, sin\frac{\theta}{2}\mathbf{v})$. Constraining $a > 0$ is equivalent to restricting the rotation angle $\theta \in [0, \pi]$. Furthermore, predicting the 8 sign categories is equivalent to predicting to which of the 8 quadrants of the 3D rotation space the $\mathbf{v}$ belongs.

3.4 Related work

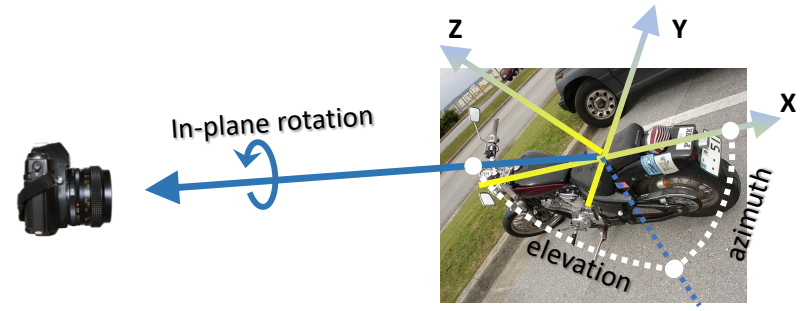
Viewpoint Estimation. In general, viewpoint estimation focuses on recovering the 3 Euler angles, namely, azimuth, elevation and in-plane rotation (see Fig. 3.3-(a)). Tulsiani and Malik [151] discretize continuous Euler angles into multiple bins and convert viewpoint estimation into a classification problem. Su *et al*. [145] propose a finer-grained discretization that divides the Euler angles into 360 bins. However, training for all possible outputs requires an enormous amount of examples that can only be addressed by synthetic renderings.

Albeit more natural, regression-based viewpoint estimation is less popular. Because of the periodical nature of angles, most approaches do not regress directly on the linear space of angles, $a, e, t \in [-\pi, \pi]$. The reason is that ignoring the angle periodicity leads to bad modeling, as the 1° and 359° angles are assumed to be the furthest apart. Instead, trigonometric representations are preferred, with [124, 8, 125] proposing to represent angles by $[cos\phi, sin\phi]$. They then learn a regression function $h : x \mapsto [cos\phi, sin\phi]$, without, however, enforcing the vectors to lie on S^1. In comparison to viewpoint classification, regression gives continuous and fine-grained angles. In practice, however, training regression for viewpoint estimation is not as easy. Complex loss functions are typically crafted, *e.g.*, smooth L_1 loss [110], without reaching the accuracy levels of classification-based alternatives.

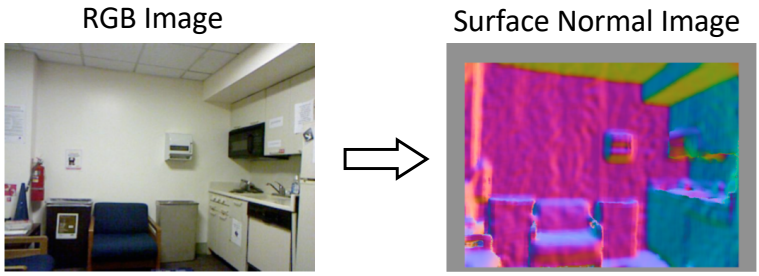
In this chapter, we continue the line of work on regression based viewpoint estimation. Built upon the S^1 representations $[cos\phi, sin\phi]$ of Euler angles [124, 8, 125], we assess our spherical regression for viewpoint prediction.

Surface Normal Estimation. Surface normal estimation is typically viewed as a 2.5D representation problem, one that carries information for the geometry of the scene, including layout, shape and even depth. The surface normal is a 3-dim vector that points outside the tangent plane of the surface. In the surface normal estimation task, given an image of a scene, a pixel-wise prediction of the surface normal is required [4, 35, 179, 42, 86, 140, 126, 156] (see Fig. 3.3-(b)).

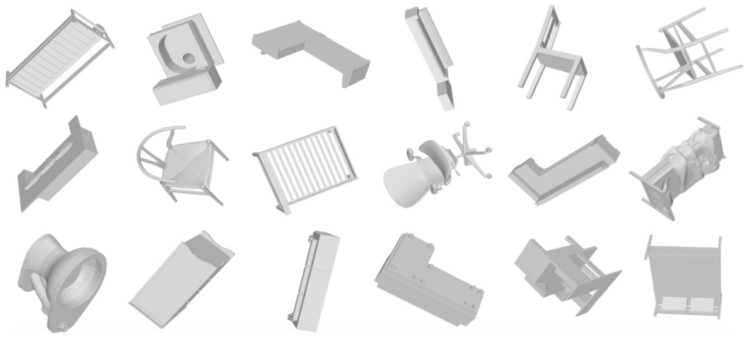
Fouhey *et al*. [42] infer the surface normal by discovering discriminative and geometrically 3D primitives from 2D images. Building on contextual and



(a) S^1: **Viewpoint** (*Pascal3D+*)



(b) S^2: **Surface Normal** (*NYU v2*)



(c) S^3: **3D rotations** (*ModelNet10-SO3*)

FIGURE 3.3: **We assess spherical regression on 3 computer vision tasks.** (a) S^1: Viewpoint estimation on Pascal3D+ [170], which needs to predict 3 Euler angles: azimuth, elevation and in-plane rotation. (b) S^2: Surface normal estimation on NYU v2 [140], where pixel-wised dense surface normal prediction is required. (c) S^3: 3D rotation on our newly proposed ModelNet10-SO3, where given one rendered view of a CAD model, we predict the underlying 3D rotation that aligns it back to standard pose.

segment-based cues, Ladicky *et al*. [86] build their surface normal regressor from local image features. They both use hand crafted features. Eigen and Fergus [35] propose a multi-scale CNN architecture adapted to predicting depth, surface normals and semantic labels. While the network outputs are

ℓ_2 normalized, the gradients are not constrained. Bansal *et al*. [4] introduce a skip-network model optimized by the standard sum of squared errors regression loss, without enforcing any structure to the output. Zhang *et al*. [179] propose to predict normals with deconvolution layers and rely on large scale synthetic data for training. Similar to [35], they also enforce an ℓ_2 norm on the output but have unconstrained gradients. Recently, Qi *et al*. [126] proposed two-stream CNNs that jointly predict depth and surface normals from a single image and also rely on the sum of squared errors loss for training.

In our work we propose a spherical exponential mapping for performing spherical regression. This new mapping can be directly applied to any of the surface normal estimation methods that rely on a regression loss on n-spheres and improve their accuracy, as we show in the experiments.

3D Rotation Estimation. 3D Rotations are a component of several tasks in computer vision and robotics, including viewpoint and pose estimation or camera relocation. The rotation matrix for 3D rotation is a 3×3 orthogonal matrix (determinant$= 1$). Direct regression on the rotation matrix via neural networks is difficult, as the output lies in the $\mathbb{R}^9$ (3×3) space. Moreover, regressing a rotation matrix directly cannot guarantee its orthogonality. Recently, Falorsi *et al*. [37] take a first step toward regressing 3D rotation matrices. Instead of predicting the 9 elements of rotation matrix directly, they pose the 3D rotation as an $S^2 \times S^2$ representation problem reducing the number of elements to regress on to a total of 6.

Viewpoint [151, 145, 110, 30, 8, 110, 114] and pose [120, 124] consider the relative 3D rotation between object and camera. With 3 consecutive rotation angles, see Fig. 3.3 (a), *Euler Angles* can uniquely recover the rotation matrix. As such a decomposition is easy to be interpreted and able to cover most of the viewpoint distribution, it has been widely adopted. However, this approach leads to the gimbal lock problem [64], where the degrees of freedom for the rotations are reduced.

Mahendran *et al*. [107] studied an axis-angle representation for viewpoint estimation by first choosing a rotation axis and then rotating along it by an angle θ. To constrain the angle $\theta \in [0, \pi)$ and the axis $v_i \in [-1, 1]$, they propose a $\pi \cdot tanh$ non-linearity. Also, instead of a standard regression loss, *e.g*. cosine proximity or sum of squared errors loss, they propose a geodesic loss which directly optimizes the 3D rotations in $\mathcal{SO}(3)$. Do *et al*. [31] consider the Lie-algebra $\mathcal{SO}(3)$ representation to learn the 3D rotation of the 6 DoF pose of an object. It is represented as $[x, y, z] \in R^3$, and can be mapped to a rotation matrix via the Rodrigues rotation formula [12]. They conclude that an ℓ_1 regression loss yields better results.

Last, both Kendall *et al*. [76] and Mahendran *et al*. [107] consider quaternion for camera re-localization and viewpoint estimation. As quaternions allow for easy interpolation and computations on the S^3 sphere, they are also widely used in graphics [139, 24] and robotics [111]. Although Do *et al*. [31]

argue that quaternion is over-parameterized, we see this as an advantage that gives us more freedom to learn rotations directly on the n-sphere.

Despite the elegance and completeness of the aforementioned works, modelling 3D rotations is hard and methods specialized for the task at hand, instead, typically reach better accuracies. Unlike most of the aforementioned works, we learn to regress on the Euclidean space directly. Furthermore, we present a framework for regressing on n-spheres with constrained gradients, leading to more stable training and good accuracy, as we show experimentally.

3.5 Experiments

3.5.1 S^1: Viewpoint estimation with Euler angles

Setup. First, we evaluate spherical regression on S^1 viewpoint estimation on Pascal3D+ [170]. Pascal3D+ contains 12 rigid object categories with bounding boxes and noisy rotation matrix annotations, obtained after manually aligning 3D models to the 2D object in the image. We follow [151, 145, 125, 108, 114] and estimate the 3 Euler angles, namely the azimuth, elevation and in-plane rotation, given the ground truth object location. A viewpoint prediction is correct when the geodesic distance $\Delta(R_{gt}, R_{pr}) = \frac{||log R_{gt}^T R_{pr}||_{\mathcal{F}}}{\sqrt{2}}$ between the predicted rotation matrix R_{pr} (constructed from the predicted Euler angles) and the ground truth rotation matrix R_{gt} is smaller than a threshold θ [151]. The evaluation metric is the accuracy $Acc@\pi/6$ given threshold $\theta = \pi/6$. We use ResNet101 as our backbone architecture, with a wider penultimate fully connected layer in the prediction head that is shared by the regression branch and classification branch (see supplementary material for details). As many of the annotations are concentrated around the x-axis, we found that rotating all annotations by 45° during training (and rotating back at test time) leads to more balanced distribution of annotations and better learning. For training data, we also use the synthetic data provided by [145], without additional data augmentations like in [107, 108].

Results. We report comparisons with the state-of-the-art in Table 3.1. Note that our spherical exponential mapping can be easily used by any of the regression-based methods with S^1 representation $[cos\phi, sin\phi]$ [124, 8]. In this experiment we combine it with Penedones *et al.* [124], who tried to directly regress 2D representation $[cos\phi, sin\phi]$ of angels, obtaining a significant improvement in accuracy over other regression and classification baselines. That said, during experiments we observed that classification-based methods are more amenable to large data sets, most probably because of their increased number of parameters. As expected, the continuous outputs by

TABLE 3.1: S^1: **Viewpoint estimation with Euler angles.** Comparison with state-of-the-art on Pascal3D+. Adding our S^1_{exp} spherical regression on top of the backbone network of [124] leads to best accuracy. We report a class-wise comparison in supplementary.

	MedErr$\downarrow$	**Acc@**$\frac{\pi}{6}\uparrow$
Mahendran *et al*. [107]	16.6	N/A
Tulsiani and Malik [151]	13.6	80.8
Mousavian *et al*. [114]	11.1	81.0
Su *et al*. [145]	11.7	82.0
Penedones *et al*. [124]†	11.6	83.6
Prokudin *et al*. [125]	12.2	83.8
Grabner *et al*. [50]	10.9	83.9
Mahendran *et al*. [108]	10.1	85.9
This chapter: [124]†+ S^1_{exp}	**9.2**	**88.2**

† Based on our implementation.

the spherical regression are better suited for finer and finer evaluations , that is $Acc@\pi/12$ and $Acc@\pi/24$ (supplementary material). We conclude that spherical regression is successful for viewpoint estimation with Euler angles.

3.5.2 S^2: Surface normal estimation

Setup. Next, we evaluate spherical regression for S^2 surface normal estimation on the NYU Depth v2 [140]. The NYU Depth v2 dataset contains 1,449 video frames of indoor scenes associated with Microsoft Kinect depth data. We use the ground truth surface normals provided by [140]. We consider all valid pixels across the whole test set during evaluation [179]. The evaluation metrics are the (*Mean* and *Median*), as well as the accuracy based metric, namely the percentage of correct predictions at given threshold *11.24°*, *22.5°* and *30°*). We implement our S^2_{exp} spherical regression based on the network proposed by Zhang *et al*. [179], which is built on top of VGG-16 convolutional layers, and a symmetric stack of deconvolution layers with skip connections for decoding. As in viewpoint estimation, we also rotate the ground truth around the z-axis by 45° to yield better results. We follow the same training setup as [179], that is we first pre-train on the selected 568K synthetic data provided by [179] for 8 epochs, and fine-tune on NYU v2 for 60 epochs.
Results. We report results in Table 3.2. Replacing regular regression in [179] with spherical regression on S^2 improves the estimation of the surface normals considerably. We found the improvement is attentuated by the fact that for surface normal estimation we perform one regression per pixel location.

TABLE 3.2: S^2: **Surface normal estimation** Comparison with state-of-the-art on NYU v2. Adding our S^2_{exp} spherical regression on top of the backbone network of Zhang *et al*. [179] leads to best accuracy.

	Mean↓	**Median**↓	11.25°↑	22.5°↑	30.0°↑
Fouhey *et al*. [42] §	37.7	34.1	14.0	32.7	44.1
Ladicky *et al*. [86] §	35.5	25.5	24.0	45.6	55.9
Wang *et al*. [156] §	28.8	17.9	35.2	57.1	65.5
Eigen and Fergus [35]	22.3	15.3	38.6	64.0	73.9
Zhang *et al*. [179]	21.7	14.8	39.4	66.3	76.1
This chapter: [179] + S^2_{exp}	**19.7**	**12.5**	**45.8**	**72.1**	**80.6**

§ Copied from [35].

TABLE 3.3: S^3: **3D Rotation estimation with quaternions.** Comparison on newly established ModelNet10-SO3. Adding our S^3_{exp} spherical regression on top of an AlexNet or VGG16 backbone network leads to best accuracy.

	MedErr↓	**Acc@**$\frac{\pi}{6}$↑	**Acc@**$\frac{\pi}{12}$↑	**Acc@**$\frac{\pi}{24}$↑
AlexNet (Direct+smooth-L1)	46.1	32.5	11.2	2.5
AlexNet + S_{flat}	33.3	53.5	34.1	13.9
AlexNet + S^3_{exp}	**25.3**	**65.4**	**48.5**	**24.4**
VGG16 (Direct+smooth-L1)	36.8	46.7	29.4	13.4
VGG16 + S_{flat}	25.9	63.5	48.7	29.5
VGG16 + S^3_{exp}	**20.3**	**70.9**	**58.9**	**38.4**

As each one of these regressions could return unstable gradients, bounding the total sum of losses with spherical regression is beneficial. Especially for the finer regression thresholds of $11.25°, 22.5°$. We conclude that spherical regression is successful also for surface normal estimation.

3.5.3 S^3: 3D Rotation estimation with quaternions

Setup. Last, we evaluate S^3_{exp} spherical regression on 3D rotation estimation on S^3 with quaternions. For this evaluation we introduce a new dataset, *ModelNet10-SO3*, composed of images of 3D synthetic renderings. ModelNet10-SO3 is based on *ModelNet10* [169], which contains 4,899 instances from 10 categories of 3D CAD models. In ModelNet10 the purpose is the classification of 3D shapes to one of the permissible CAD object categories. With ModelNet10-SO3 we have a different purpose, we want to evaluate 3D shape

alignment by predicting its 3D rotation matrix w.r.t. the reference position from single image. We construct ModelNet10-SO3 by uniformly sampling per CAD model 100 3D rotations on $\mathcal{SO}(3)$ for the training set and 4 3D rotations for the test set. We render each view with white background, thus the foreground shows the rotated rendering only. We show some examples in Fig. 3.3-(c).

Relying on Euler angles for ModelNet10-SO3 is not advised because of the Gimbal lock problem [64]. Instead, alignment is possible only by predicting the quaternion representation of the 3D rotation matrix. For this task, we test the following 3 regression strategies:

(I) Direct regression with smooth-L1 loss. It may cause the output to no longer follow unit ℓ_2 norm.

(II) Regression with ℓ_2 normalization $\mathcal{S}_{flat}$.

(III) Regression with $\mathcal{S}_{exp}$ (this chapter).

We report results based on AlexNet and VGG16 as our CNN backbones, with a class-specific prediction head. We borrow the evaluation metric from viewpoint estimation, namely *MedErr* and *Acc*@$\{\pi/6, \pi/12, \pi/24\}$ so that we also examine finer-grained predictions.

Results. We report results in Table 3.3. First, both S_{flat} and S^3_{exp} regression on quaternions improve over direct regression baselines. This shows the importance of constraining the output space to be on a sphere when regress spherical target. Second, putting l_2 normalization constraint on output space, S^3_{exp} improves over S_{flat} with both AlexNet and VGG16. For AlexNet we obtain about $8-12\%$ improvement across all metrics. VGG16 is higher overall, but the improvement over the baseline is less. This shows that with the VGG16 we are potentially getting closer to the maximum possible accuracy attainable for this hard task. That can be explained by the fact that the shapes have no texture. Thus, a regular VGG16 is close to what can be encoded by a good RGB-based model. Note that estimating the 3D rotation with a discretization and classification approach [151, 145, 110, 30] would be impossible because of the vastness of the output space on $\mathcal{SO}(3)$ manifold.

Further, we investigate the variance of the gradients $\frac{\partial \mathcal{L}}{\partial O}$ by recording its average ℓ_2 norm during training progress. The results are shown in Fig. 3.4. We observe the gradient norm of the spherical exponential mapping has much lower variance. Spherical exponentiation achieves this behavior naturally without interventions, unlike other tricks (*e.g.* gradient clipping, gradient reparameterization) which fix the symptom (gradient instability/vanishing/exploding) but not the root cause (unconstrained input signals). We conclude that spherical regression is successful also for the application of 3D rotation estimation.

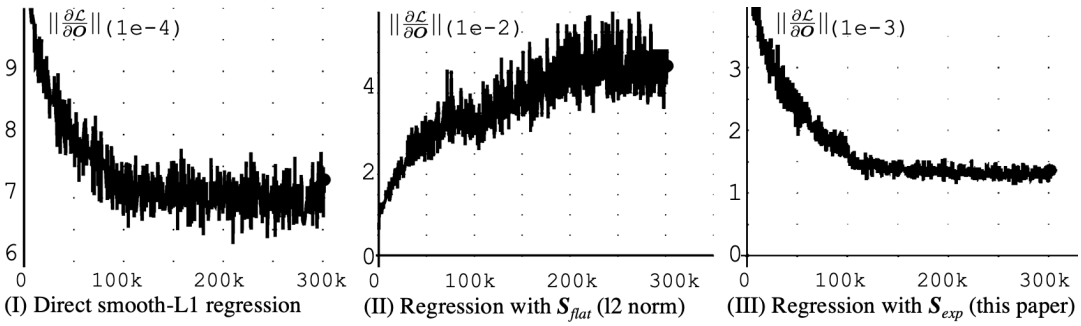


FIGURE 3.4: **Variance of the average gradient norm $||\frac{\partial \mathcal{L}}{\partial O}||$.** Spherical exponentiation $\mathcal{S}_{exp}$ yields lower variance on mini-batch over entire train progress.

3.6 Conclusion

Spherical regression is a general framework which can be applied to any continuous output problem that lives in n-spheres. It obtains regression gradients that are constrained and well-behaving for several computer vision challenges. In this work we have investigated three such applications, specifically viewpoint estimation, surface normal estimation and 3D rotation estimation. Generally, we observe that spherical regression improves considerably the regression accuracy in all tasks and different datasets. We conclude that spherical regression is a good alternative for tasks where continuous output prediction are needed.

Part II

Deep learning with label geometry

Chapter 4

Quasibinary Classifiers for Image Classification

4.1 Introduction

Learning deep convolutional neural networks to classify an image requires a proper prediction activation function. It maps an internal network representation from feature space to prediction space, where a loss can be more easily defined. Softmax has been the dominant choice, and very successfully so [83, 146, 58, 66]. In general, the softmax classifier is designed to model the probabilities for *one-vs.-rest* classification problems, where the number of ground truth labels per sample is assumed to be one. As the output predictions are normalized to be summed to one, softmax becomes a natural fit to return the categorical distribution prediction. However, this one-label assumption limits the application of softmax for problems where the number of labels for a sample is not one, most notably zero-label [92, 88] and multi-label [127, 49, 91] classification problems. Nonetheless, the softmax classifier is sometimes still being considered for zero-label [61, 60] and multi-label [127, 173, 91] classification problems, with the risk of violating the categorical assumption.

Instead of 'abusing' the softmax for out-of-distribution and multi-label problems, one may opt to build a set of binary classifiers for each of the classes, *e.g.* [116, 87, 115, 162, 163, 18]. However, binary classifiers suffer from modeling class likelihoods independently. As a result, building binary classifiers becomes a suboptimal choice when the number of classes is large or imbalanced [116, 87, 138, 162]. Moreover, given an input image, the confidence scores per classifier are uncalibrated, making them incomparable in practice.

In this chapter, we introduce a new definition of binary classifiers, which we coin *quasibinary classifiers*. Similar to regular binary classifiers, quasibinary classifiers compute probabilities for binary outputs. Yet, different from regular binary classifiers, quasibinary classifiers incorporate the information that other labels may exist in the image. We achieve this by having the normalization function of the quasibinary classifiers learnt rather than defined,

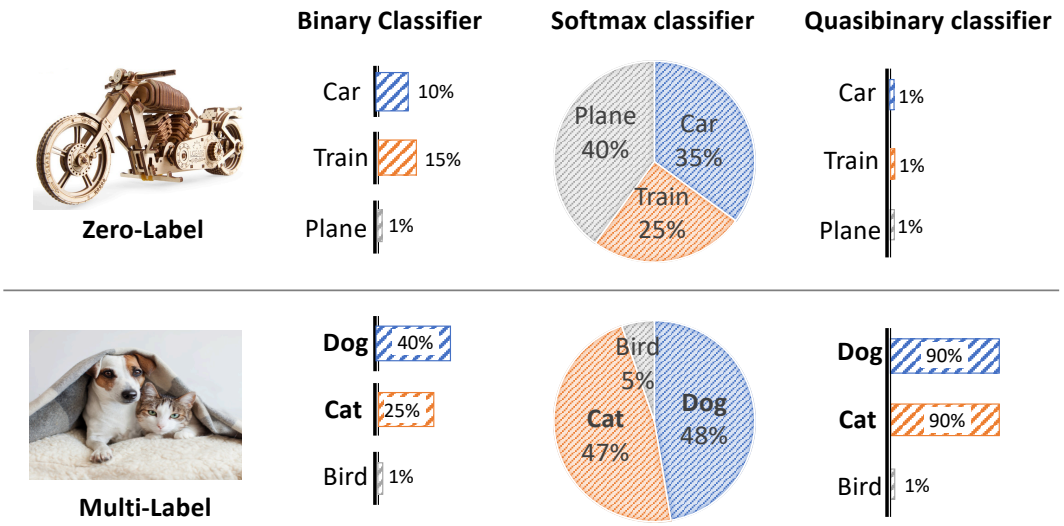


FIGURE 4.1: **Comparison of quasibinary classifiers with binary and softmax classifiers for zero-label and multi-label classification.** While binary classifiers can in principle handle both cases, the confidence scores are not credible. Softmax fails in both cases as it restricts the sum of all the confidence to be equal to one. Our proposed quasibinary classifiers are trained jointly and assign credible confidence scores in both cases.

as with binary and softmax classifiers. Specifically, quasibinary classifiers set the normalization function to be a constant and share it not only between classes but also between data points, allowing for tractability, high efficiency, and most importantly better calibration in computing the probabilities. As a result, quasibinary classifiers can work seamlessly in a variety of classification settings without the need for any specialized adaptation. They work well in regular single-label, multiple-class settings, as well as in multiple-label multiple class settings. They even work well when none of the labels are present in the image, that is out-of-distribution classification [61, 163] where the out-of-distribution samples are visible during training, but without any label attached to them. Importantly, quasibinary classifiers yield calibrated probabilities that can be used to compare predictions more reliably amongst different classes and different images, see Fig. 4.1.

We make the following contributions. First, we identify restrictions of the widely used binary and softmax classifiers. Second, on the basis of these restrictions, we propose quasibinary classifiers that learn a constant normalization function shared among classes and data points to return well calibrated binary outputs. Third, we show that quasibinary classifiers perform well on a variety of classification settings that are important in realistic application scenarios, including multiple- or even zero-label out-of-distribution image classification.

4.2 Background

Thanks to deep learning, probabilistic learning has become the *de facto* choice for training classifiers [83, 146, 58, 66]. Probabilistic classifiers, be it binary or multi-class, maximize the (log-) likelihood $p(Y|X)$ on the training data $\{X, Y\} = \{(x^{(i)}, y^{(i)})\}$. Specifically, both binary and multi-class classifiers assume the following decomposition of the total probability

$$p(Y|X) = \prod_i p(y^{(i)}|x^{(i)}) \tag{4.1}$$

The decomposition in equation (4.1) is based on the assumption that the class predictions $y^{(i)}$ are conditionally independent given the input data points $x^{(i)}$. The classifiers are then typically the neurons $p_k, k = 1, ..., K$ for K classes in the ultimate layer of the neural network. For probabilistic training of the classifiers the neurons p_k must correspond to valid probabilities, namely, $p_k \in [0, 1]$. In a probabilistic, rather than a frequentist perspective, this probability can also represent the *belief* that one has in the event k being true or not.

4.2.1 Ensembles of sigmoid classifiers

For binary classifiers given K classes, the random variables $y_1^{(i)} \in \{0, 1\}, ..., y_K^{(i)} \in \{0, 1\}$ of the i-th sample correspond to the K independent predictions. These K random variables follow a Bernoulli distribution, $y_k^{(i)} \sim \text{Bernoulli}(p_k^{(i)})$, which means p_k can be conveniently modelled by independent sigmoids, that is

$$p_k^{(i)} = p(y_k^{(i)}|x^{(i)}) = \frac{\exp(z_k^{(i)})}{1 + \exp(z_k^{(i)})} \tag{4.2}$$

where $z_k^{(i)} = h(x^{(i)})$ is the k-th logit computed by the neural network $h(\cdot)$ on the input $x^{(i)}$. The probability output space of the Bernoulli random variable is complemented by $\bar{p}_k$, such that $\bar{p}_k = 1 - p_k$. This is guaranteed by the normalization factor $C_k^{(i)} = 1 + \exp(z_k^{(i)})$. Since binary classifiers work independently for each class given a data sample, it is possible to use the binary classifier in multi-label classification settings. As binary classifiers do not consider the dependency between classes, their performance is usually sub-optimal [116, 87, 138, 162].

We note that while sigmoid activation functions have been conflated with binary and independent classifiers, their function is simply to return binary

decisions. Being independent is a modeling choice. This is important to emphasize in the context of the proposed quasibinary classifiers that we introduce later.

4.2.2 Softmax classifiers

When having multiple classes in the training set while knowing that each image contains only a single class, *i.e.*, $\#label = 1$, the classifier neurons are usually parameterized by softmax functions. The softmax function models the probability of a categorical distribution parameterized by the means

$$p_k^{(i)} = p(y_k^{(i)}|x^{(i)}; \#\text{label=1}) = \frac{\exp(z_k^{(i)})}{\sum_j \exp(z_j^{(i)})}. \tag{4.3}$$

We observe that both definitions of the binary classifier and the softmax classifier in equations (4.2) and (4.3) have a similar form. The numerator is precisely the same and equal to the exponentiation of the logit $z_k^{(i)}$. We refer to this as the scoring function $s_k^{(i)} = \exp(z_k^{(i)})$. The denominator, however, is different. While in the binary classifier the denominator only depends on the same logit $z_k^{(i)}$, in the softmax classifier the denominator depends on the logits from all classes. This is beneficial for optimization, as using the logits from all classes couples together the otherwise independent scoring functions and trains them jointly.

Despite the popularity and accuracy of softmax over the binary classifier, however, the softmax classifier also suffers from a limitation: the number of labels is not always known at test time. Most notable for out-of-distribution data (#label=0) and multiple-label data (#label=n). In both cases, the probability outputs are meaningless.

In this work, we are interested in defining a classifier that inherits the flexibility of binary classifiers, while leveraging the prior knowledge of the number of labels per sample that is given for free for optimization. To this end, we present *quasibinary classifiers*. Quasibinary classifiers seamlessly handle any-label (even zero-label) classifications like a binary classifier, but they can be trained in a coupled way like softmax. They also return much more calibrated probabilities as per the definitions recently introduced by Guo *et al*. [51].

4.3 Quasibinary classifiers

4.3.1 Definition

As with the ensembles of binary classifiers, we want binary decisions since in the image there can be multiple classes present. That is, we have again random variables $y_1^{(i)} \in \{0,1\}, ..., y_K^{(i)} \in \{0,1\}$ mapped to the binary output space. Different from the ensembles of binary classifiers, we do not assume that these binary classifiers are independent to each other. That is, we want the likelihood terms of the random variables to be both binary and correlated.

In defining our model, we also start from Bernoulli random variables. However, in the mean parameters of the Bernoulli distributions we introduce a shared constant C that is not a random variable and is shared across all likelihood terms across classes (like softmax) and -importantly- all images. Namely, our Bernoulli variables are modelled as $y_k^{(i)} \sim Bernoulli(q_k)$, where

$$q_k^{(i)} = q(y_k^{(i)}|x^{(i)}, X_{\setminus(i)}) = \frac{\exp(z_k^{(i)})}{C(x^{(i)}, X_{\setminus(i)})} = \frac{\exp(z_k^{(i)})}{C(X)}. \tag{4.4}$$

Note that, as intended, the normalization constant is shared between samples and does not depend on specific classes. Therefore, the predictions modelled by the Bernoulli random variables are still binary but not independent. As such, quasibinary classifiers can seamlessly work in multiple-label or zero-label setting, where an image may contain several objects from different classes or none at all. In the case of multiple labels present in an image, each of the Bernoulli variables shall return their confidence in the class being present or not, albeit these confidences are not independent to each other. And in the case the image contains no relevant objects, the Bernoulli variables shall all return low confidence without being forced to either select wrongly one of the classes as being present or to assign the same likelihood to all wrong classes.

For the quasibinary classifiers we use the same scoring function as the binary and softmax classifiers, that is an exponential $\exp(\cdot)$, which ensures positivity. Importantly, note that now the normalization function depends on all the training data, $C(X)$. Namely, we have a normalization function that is *shared* across all classes and training data points, as it depends on the logits of all the training data points. This is crucial, since *(i)* a shared normalization function across classes is able to couple individual binary classifiers together, thus jointly optimizing them while taking into account the knowledge of how many labels are present. Moreover, *(ii)* having a shared normalization function across training data points allows the quasibinary scoring functions to "communicate" with each other. Thus, the classifiers learn to predict confidence scores that are comparable across classes and different data points,

thus returning better calibrated probabilities.

As with ensembles of binary classifiers, the probability output space is complemented by $\bar{p}_k = 1 - p_k$, thus having the total sum $\bar{p}_k + p_k = 1$, which is a requirement for a valid probability space. Furthermore, for a valid probability space we need to make sure that $0 < p_k < 1$, which is possible by a careful choice of an activation function for the classifier neurons. In the end, the joint likelihood is equal to

$$p(Y|X) = \prod_i p(y^{(i)}|X) = \prod_i p(y^{(i)}|x^{(i)}, X_{\setminus(i)}), \tag{4.5}$$

where the conditioning variables $x^{(i)}, X_{\setminus(i)})$ together make up for the X variable in the normalization constant $C(X)$.

Choosing normalization function $C(X)$. Clearly, the choice of the normalization function is critical. There exist several requirements.

First, we want our quasibinary classifiers to be jointly optimized while exploiting the free prior knowledge of the number of labels per training sample, *i.e.* #label $= n$ where $n \in [0, 1, \cdots, K]$. We can show that this prior knowledge is equivalent to the following constraint in the predicted probability q_k,

$$\sum_k q_k = \text{\#label}. \tag{4.6}$$

We provide the proof in the supplementary material. Note that the reason for the sum of all q_k being possibly greater than 1 is that the presence (or not) of the various labels y_k in a training sample is not mutually exclusive. As a special case, consider #label=1, equation (4.6) is exactly what the softmax classifier enforces. A second requirement regarding the normalizing function is computational tractability, as depending on all data, $x^{(i)}, X_{\setminus(i)}$, is potentially prohibitive.

4.3.2 Algorithm

In the previous section we explained that the normalization function of quasibinary classifiers is a function shared across classes and across data points. As this normalization function is shared across all data points and classes, it needs to be a constant function. This constant normalization function must then be learned throughout the training so as to satisfy several constraints. First, by the end of the training the probability estimates must lie within the $[0, 1]$ range. Second, the normalization function must be trained to adhere to the constraint of equation (4.6). That is, we want our quasibinary classifiers

to satisfy

$$\arg\max \sum_{i,k} y_k^{(i)} \log(q_k^{(i)}) \tag{4.7}$$

$$\text{s.t. } q_k^{(i)} \in [0,1] \quad \forall i,k \tag{4.8}$$

$$\sum_k q_k^{(i)} = \#\text{labels}^{(i)} \tag{4.9}$$

Similar to stochastic gradient descent, we rely on stochastic mini-batches B instead of the whole training set.

Training. Rather than enforcing the constraint in equation (4.6) for all samples, we relax the constraint to batch level for computational tractability. Namely, given that our normalizing function is a constant we have that

$$\sum_i^B \sum_k^K \exp(z_k^{(i)})/C = \sum_i \#\text{label}^{(i)}$$

$$\Rightarrow \quad C = \frac{1}{N} \sum_{i=1}^B \sum_{k=1}^K \exp(z_k^{(i)}) \tag{4.10}$$

where N is the total number of labels a batch of data has. Following [84] we resort to Lagrange relaxation to derive the final learning objective and optimize for the model parameters θ

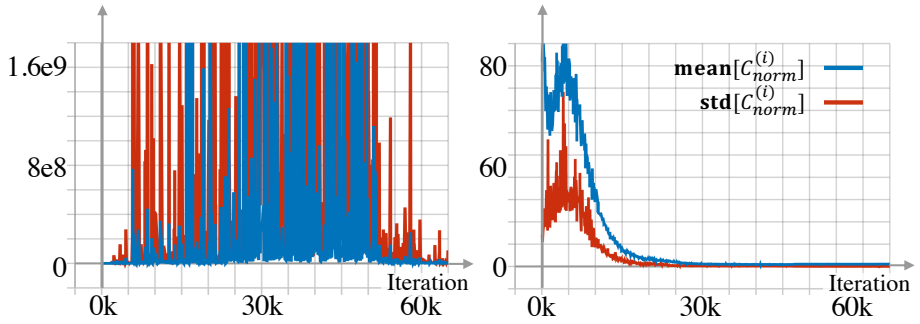
$$\mathcal{L}(\theta) = \frac{1}{B} \sum_{i=1}^B \sum_{k=1}^K y_k^{(i)} \cdot \log q_k^{(i)} - \max(\log q_k^{(i)}, 0), \tag{4.11}$$

$$\text{where } \log q_k^{(i)} = z_k^{(i)} - \log C \tag{4.12}$$

The summands correspond to the losses by the q_k likelihood terms. The second term makes sure that q_k yields by the end of the training a valid probability (in log-space the maximum probability is 0).

Note that because of the Lagrange relaxation, it is *not theoretically guaranteed* that q_k will always be within the $(0,1)$ range and thus yield a probability. However, we find the optimizer is able to find good enough solution to satisfy the constraint. In practice, we only observed a negligible amount ($\sim$ 0.1%) of violations *i.e.* ($q_k > 1$) on test data, which are simply clipped to [0,1] to makes sure q_k are proper probabilities. Similar optimizations for learning to compute probabilities were also previously proposed in [28] with success. Note also that during training time as the model gets updated per iteration, C varies till convergence.

Training in batches. It is important to note that we define the normalization function in equation (4.10) as a batch-level implementation of equation (4.6). This means that the number of labels changes per image. Also, just



(a) Softmax (b) Quasibinary

FIGURE 4.2: **The normalization function $C(X)$ converges over time to a constant value on CIFAR100.** We observe that the mean and variance of the softmax normalization are up to 8x larger than for the quasibinary normalization. As we can see in (a), the variance is extreme due to alternating spikes (we only show a cropped range between 0 and 1.6e9). The fluctuations continue with no convergence. That is expected, as for softmax the normalization needs not to converge for accurate classification. However, a fluctuating normalizing constant means that scores between different images are hardly comparable. Overall, softmax behaves completely opposite to the quasibinary.

like other multi-label classification settings, the total number of classes K is fixed both at training and test. As the training is in batches, with imbalanced datasets it is important to account for the class frequencies in the batch constitution. We simply follow the spirit of SGD to randomly select a mini-batch of samples [49, 173, 18].

Testing. After training we substitute the normalization function $C(X)$ in equation (4.4) with the moving average of $C^{(\text{mavg})}$ as a constant to make prediction at test time. Thus, unlike softmax classifiers that assume #label=1 on test data to compute normalization factor, we do not need to make any such assumptions.

Moving average of normalization constant. Since it is inefficient to compute at test time the normalization factor C based on training data, we track a moving average of $C^{(t)}$ over training iterations,

$$C^{(\text{mavg})} = (1-\alpha)\cdot C^{(\text{mavg})} + \alpha \cdot C^{(t)}, \tag{4.13}$$

where $C^{(t)}$ is the constant normalization function at iteration t, see Fig 4.2. This is similar to batch-normalization. Note that just like in batch normalization, the α smooths out the fluctuations and helps the learnt constant to converge to a single value consistent for all images, as shown in Fig. 4.2. Further, we empirically find that the training is not negatively affected by the

moving average computation in terms of accuracy. We find that the algorithm is rather robust in the choice of α with the differences in the variance of $C^{(\text{mavg})}$ up to 1e-2. We find that setting $\alpha = 0.1$ is good enough.

4.4 Related work

4.4.1 Zero-label problems

In general, zero-label data refers to irrelevant samples that belong to neither class of the in-distribution training data. Recent deep models achieve good accuracy on in-distribution data, but are known to be over-confident on out-of-distribution data [61, 60]. Based on statistics of the softmax prediction, Hendrycks and Gimpel [61] introduce a baseline detector to differentiate misclassified samples from out-of-distribution samples. They point out the softmax probabilities have a poor direct correspondence to model confidence, but do not provide a solution. Liang *et al*. [92] find that using temperature scaling and adding small perturbations to the input data increases the statistical significance of the softmax output for the in- and out-of-distribution samples. This leads to an improvement in detection performance.

Passively detecting zero-label out-of-distribution data based on a trained model is not enough. Rather, one may opt to build a model that is aware of out-of-distribution data during training. To do so, Lee *et al*. [88] add two additional loss terms upon the standard cross-entropy loss to train a softmax classifier. The first loss models out-of-distribution data by enforcing the softmax to output a uniform distribution prediction. The second loss guides a generative adversarial network to generate the most effective out-of-distribution samples for training. In contrast, Hein *et al*. [60] introduce noise data as out-of-distribution samples during training. With the same intention as Lee *et al*. [88] to encourage an uniform distribution prediction for out-of-distribution data, Hein *et al*. [60] propose a loss function that suppresses the largest predicted confidence from softmax.

In this chapter, we continue the line of work on explicitly modeling out-of-distribution data. However, instead of introducing new losses for the softmax classifier, we introduce the quasibinary classifier which is able to handle out-of-distribution data by design.

4.4.2 Multi-label problems

In most real life classification problems, one sample may be associated with multiple labels at the same time, rather than just one. For example, an image from a social network can have multiple user tags and a medical image may show none or multiple symptoms of a certain diseases. Multi-label classification considers problems where the number of labels for a sample is

unknown. The traditional strategy is to reduce the multi-label classification problem to multiple binary classification problems [115, 162, 163, 18]. When each class is being modeled independently, an ensemble of binary classifiers is able to make multi-label predictions. However, as binary classifiers are trained independently, their confidence scores lack calibration. As a result, binary classifiers have difficulty when extended to multi-label problems with a large number of classes.

Although softmax is exclusively designed for one-vs.-rest classification, it is also being adapted to solve multi-label classification [127]. For example, the softmax with cross-entropy loss is trained to push the prediction of a multi-label sample to be an equally weighted multi-hot vector. However, as softmax forces the sum of the output predictions to be one, a softmax classifier cannot provide credible confidence scores for multi-label classification problems. Alternatively, one may transform a multi-label classification problem into a ranking problem [165, 49, 178, 91, 168]. Given training data, the objective is to learn a model able to rank the most relevant labels above the irrelevant ones. However, as such a strategy does not directly solve a classification problem it cannot provide model confidence of the predictions. In this chapter, we are interested in building a multi-label classifier that can provide reasonable model confidence scores.

4.5 Experiments

In this section, we evaluate the quasibinary classifier on several classification problems, with a variety of benchmark datasets and a Resnet18 [57] backbone. Although the evaluation settings may depend on the specific problem, the training settings are mostly the same. To avoid repeated description, we first detail this common training setting for all models.

Common training protocol. All model parameters are randomly initialized, following He *et al*. [58]. We use the SGD optimizer with momentum set to 1e-4. The initial learning rate is set as 0.1 and decreased by a factor of 10 after 50% and 75% of the epochs. We train all the models for 200 epochs with a batch size of 64. For the quasibinary classifier, we set the momentum of the moving average used to track the normalization factor during training as 0.9.

4.5.1 One-vs.-rest image classification

Setup. First, we evaluate the performance of the quasibinary classifier on the common image classification problem with one ground truth label per image. While we are particularly interested in the comparison with the binary classifier, we also compare with the softmax classifier, known to be the better solution for this problem. We choose four datasets, *i.e*. ***CIFAR10*** [82], ***CIFAR100*** [82], ***Tiny-ImageNet*** and ***ImageNet*** [27] with {10, 100, 200, 1000}

TABLE 4.1: **One-vs.-rest image classification.** Comparison of top-1 error rate on CIFAR10, CIFAR100, Tiny ImageNet and ImageNet, with the total number of classes being 10, 100, 200 and 1000. Binary classifiers are good for small amounts of classes, but have difficulty to converge as the number of classes increases. In that case our quasibinary classifier does much better. For up to 200 classes it is even comparable with the softmax classifier.

	CIFAR10	**CIFAR100**	**Tiny-ImageNet**	**ImageNet**
Binary classifier	**4.8**	35.4	×	×
Quasibinary classifier (*Ours*)	4.9	**21.9**	**42.9**	25.4
Softmax classifier	5.2	22.2	43.3	**23.9**

classes respectively. The total number of images for each of the three datasets are 60K, 60K, 110K, and 1200K, and for all datasets the images are equally distributed over each class. Except for the ImageNet experiments where we use standard 224×224 input size, we always use a 32×32 input size in order to share the same network architecture. The top-1 error rate on the test sets is reported.

One-vs.rest image classification results. We report results with models trained with a Resnet18 [57] backbone in Table B.1. Similar results were obtained with VGG16 and DenseNets, see supplementary material. The performance differences between the binary classifier and our quasibinary classifier are subtle on CIFAR10, when there are only a small number of classes. Already for 100 classes, on CIFAR100, our quasibinary classifier does much better. When increasing the number of classes further the binary classifier fails to converge. This is partially due to the fact that each binary classifier models the likelihood prediction independently. We further suspect the optimization difficulty of binary classifiers on large class dataset is due to the sigmoid activation function, which is known to saturate easily [48]. Interestingly, the quasibinary classifier is even competitive with the softmax classifier for classification problems up to 200 classes. For the more challenging ImageNet setting with 1000 classes, the softmax classifier performs better, as expected.

4.5.2 Zero-label image classification

Setup. Next, we evaluate the performance of the quasibinary classifier in modeling zero-label image data. In particular, our goal is to build a model that is able to separate out data samples that belong to neither of the predefined classes (out-of-distribution classes), while maintaining the classification performance on the in-distribution classes. Similar to [88, 60], we use

out-of-distribution data for training. Previous works [61, 92, 88, 60] construct out-of-distribution samples by either taking natural image samples from other datasets different from the training source or by synthesizing noise images. As those out-of-distribution samples are usually easy to distinguish from the source dataset, and consequently most methods achieve near perfect performance, we construct a new dataset based on CIFAR100.

In ***CIFAR60+40*** the zero-label out-of-distribution data is created to have more confusion with the in-distribution data. The original CIFAR100 dataset has 20 coarse classes where each of them is subdivided into 5 fine-grained classes. Thus, a total of 100 fine-grained classes are considered in CIFAR100. We take the identical image data from CIFAR100 for CIFAR60+40, however, for each coarse class, 2 out of the 5 fine-grained classes the labels are removed, meaning they become our zero-label out-of-distribution samples. As a result, images for 60 classes are retained as labeled in-distribution (IN) and the images from the remaining 40 classes are treated as zero-label out-of-distribution (OUT). Splits will be made available.

Baselines. We compare the quasibinary classifier with three methods for modeling out-of-distribution data: 1) binary classifiers; 2) softmax classifier with $\mathcal{L}_{\text{Uniform}}$ loss [88] that minimizes KL-divergence of predicted probability distribution for OUT samples w.r.t. a uniform distribution; and 3) softmax classifier with $\mathcal{L}_{\text{MaxConf}}$ loss [60] that suppresses the predicted probability dimension with the maximum confidence for OUT samples.

Evaluation. We consider three metrics: 1) Accuracy on IN samples, 2) Mean of Maximum Confidence (MMC) [60] for OUT samples and 3) AUROC measure to evaluate how good a model can differentiate OUT from IN.

Results. We report results in Table 4.2. The quasibinary classifier and softmax with $\mathcal{L}_{\text{Uniform}}$ loss [88] obtain a similar accuracy on IN samples, but softmax with $\mathcal{L}_{\text{Uniform}}$ loss fails to suppress the MMC for OUT samples. Compared to the softmax classifier with $\mathcal{L}_{\text{MaxConf}}$ loss [60], the quasibinary classifier obtains a slightly better MMC performance on OUT samples, but a much better accuracy is achieved on IN samples (80.6% vs 45.2%). What is more, since [88, 60] optimize the softmax classifier towards a uniform distribution prediction for OUT samples, they both have a theoretical upper bound for MMC performance that is equal to 1/#classes. In theory, the quasibinary classifier and binary classifier can both assign 0% confidence to OUT data. However, in practice, we observe binary classifiers obtain a suboptimal performance on the accuracy for IN data and MMC for OUT data. We conclude our quasibinary classifier is successful in modeling zero-label samples.

4.5.3 Multi-label image classification

Setup. Last, we evaluate the quasibinary classifier for multi-label image classification on ***NUS-WIDE*** [21] and ***MS-COCO*** [97]. NUS-WIDE consist of

TABLE 4.2: **Zero-label image classification** on CIFAR60+40. Binary classifiers perform reasonable, but do not excel. Softmax based methods obtain good performance on either in-distribution accuracy [88] or out-of-distribution MMC [60], but cannot do both well. Our quasibinary classifier achieves good performance on all measures.

	IN	**OUT**	**BOTH**
	Accuracy ↑	*MMC* ↓	*AU-ROC*↑
Binary classifiers [128, 18, 162]	77.8 %	14.7 %	0.901
Softmax + $\mathcal{L}_{\text{Uniform}}$ [88]	**80.7** %	59.8 %	0.800
Softmax + $\mathcal{L}_{\text{MaxConf}}$ [60]	45.2 %	7.4 %	0.764
Quasibinary classifier	80.6 %	**6.9** %	**0.913**

TABLE 4.3: **Multi-label image classification.** Although the softmax returns good predictions in multi-label settings, their calibrated confidence scores (ECE) suffer compared to the proposed quasibinary classifiers, even when softened with a temperature. This indicates that softmax classifiers make unjustifiably overconfident predictions.

	MS-COCO		**NUS-WIDE**	
	F_1↑	ECE(%)↓	F_1↑	ECE(%)↓
Binary classifier [128, 18, 162]	51.2	26.8	40.7	23.6
Softmax [106]	**54.7**	32.2	43.2	25.8
Softmax w/ temperature [62]	**54.7**	31.4	43.2	24.6
Quasibinary classifier	**54.7**	**2.8**	**43.5**	**3.3**

260K Flickr images with 81 classes selected from high frequency user tags. MS-COCO consists of 120K images that are labeled by 80 object classes. The average number of labels per image for the two datasets are 2.41 and 2.94, respectively. We follow Gong *et al*. [49] to remove 60K invalid images in NUS-WIDE and split the rest into a 150K training set and a 50K test set. For MS-COCO, we use the original training/validation splits, namely 82K for training and 40K for testing. We compare three classifiers: 1) binary classifiers [128, 18, 162], 2) Softmax classifier [127, 173, 91] and 3) quasibinary classifier. We follow the setting of Li *et al*. [91] to fine-tune a VGG16 [143] backbone for 20 epochs. During test time, the top-4 most confident predictions are outputted as suggested by Li *et al*. [91].

Evaluation. For evaluation, we first follow the conventional evaluation protocol to report macro F_1 score, which assesses the quality of the confidence score for ranking. However, we are more interested in how credible

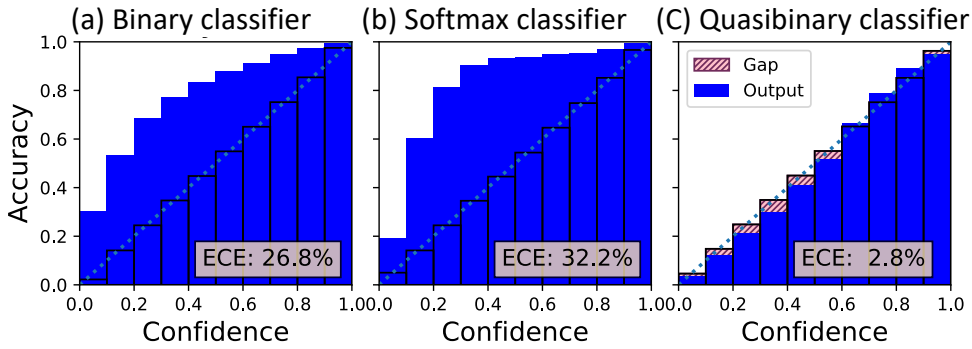


FIGURE 4.3: **Reliability diagrams for multi-label classification** on MS-COCO, where being diagonal means a perfect calibrated confidence prediction (with ECE=0). Our quasibinary classifier achieves the best confidence calibration, leading to more credible confidence scores.

the predicted confidence scores are. Thus, we also report the Expected Calibration Error (ECE) [51], which indicates how well the predicted confidence score indicates the actual likelihood of the model to make a correct prediction. Notice that ECE is mostly used to evaluate the reliability of the top-1 confidence in single-label classification problem. In our setting, we evaluate the top-4 confidence reliability.

Results. We report results in Table 4.3. Regarding the F_1 score, the performance of the quasibinary classifier and softmax are similar, while the binary classifier is worse. This is mainly because binary classifiers do not use the *#labels* as a prior during training. Considering the ECE score, we observe the quasibinary classifier improves the reliability of the confidence predictions over binary classifiers and softmax considerably. Further, we show in Fig. 4.3 the 10-bins reliability diagrams [51] as a visualization of the ECE score, where being diagonal means the accuracy of test samples in each bin aligns with received confidence score, *i.e.* perfect calibrated confidence prediction (with ECE=0). We again observe the quasibinary classifier achieves the best performance. Although modulating the temperature for softmax helps with overconfident scores in one-vs-rest classification (data not shown), when multiple labels are present the returned scores are still overconfident. The reason is that even with multiple labels the sum of outputs remains 1 and more than one logits must still share the "max" score. Last, we show qualitative results in Fig. 4.4. We observe that only the quasibinary classifier is able to assign high confidence to multiple correct labels at the same time. Most interestingly, we observe quasibinary classifier helps to retrieve part of the missing annotations with high confidence, e.g. "sunset" and "clouds" for the second image of NUS-WIDE examples. We conclude that the quasibinary classifier models multi-label classification problems with credible confidence scores.

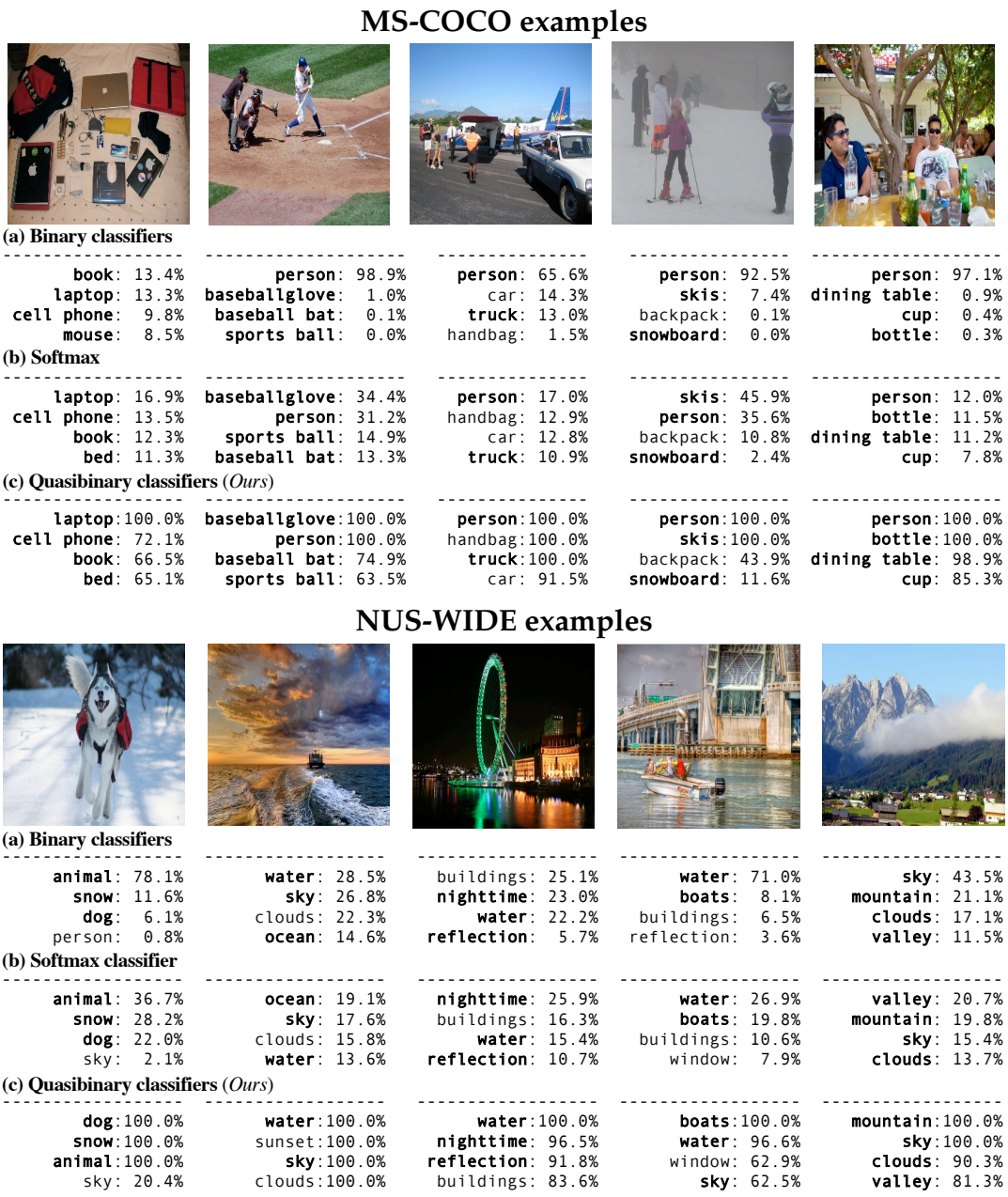


FIGURE 4.4: **Multi-label image classification with credible confidence.** We show the top-4 most confident predictions and scores for all three classifiers, with the correct prediction being marked in bold font. The images are from NUSWIDE and MS-COCO. While binary classifiers perform sub-optimal and softmax has to enforce the predictions to be summed to one, the quasibinary classifier provides credible confidence scores by design.

4.6 Conclusion

Softmax and binary classifiers face difficulties in image classification settings beyond the regular multi-class classification. Characteristic examples are multiple class, multiple label problems, where an image may contain more than one object. Another example is the zero-label out-of-distribution problem, where the image may contain none of the relevant labels. To address the limitations of binary and softmax classifers, we introduce the quasibinary classifiers. Quasibinary classifiers define a novel normalization function that is learnable, constant, and shared between classes and data points. This allows them to compute probabilities that are better calibrated and, thus, more directly comparable between classes as well other data points. We show in a variety of settings and datasets that quasibinary classifiers are considerably better in image classification settings where regular binary and softmax classifiers suffer, including zero-label and multi-label image classification. Importantly, we show that quasibinary classifiers yield well calibrated probabilities allowing for direct and reliable comparisons not only between classes but also between data points.

Chapter 5

Vec2Bundle: Learning Class Hierarchies

5.1 Introduction

While image classification has progressed in the last decade with leaps and bounds, it is natural that with an ever-increasing number of finer and finer classes more mistakes are inevitable. A root cause is the increasing visual similarity as we move towards more fine-grained categorizations, as well as the long-tail problem [102, 182, 121, 153] in collecting enough training examples for infrequently appearing classes. When classifying images, however, we are not necessarily interested in the finest of categorizations if this increases the chances of a misclassification. For instance, we are often content with recognizing an "eagle" or even "bird" instead of a "Montagu's Harrier", rather than confuse that bird's image with an "airplane". To accommodate this a simple solution is to make predictions in bundles, where bundles contain confused classes without further differentiation. As we cannot know in advance which classes get confused or even what is the optimal way to organize them in bundles, a common way is to group classes in terms of hierarchy. Specifically, in this work, we want to learn class hierarchies from data so that the resulting bundles balance between minimizing classification mistakes while avoiding trivially true predictions.

Balancing accuracy and specificity is commonly achieved by semantic hierarchies. However, semantic hierarchies are not always available for our task at hand. Moreover, when available the primary goals of semantic hierarchies are unaligned with the goals of visual categorization. The first goal is to ensure each node has a semantic interpretation. The second goal is to divide each node into mutually exclusive children nodes, or groups, till the finest possible categorization. From the standpoint of visual classification,

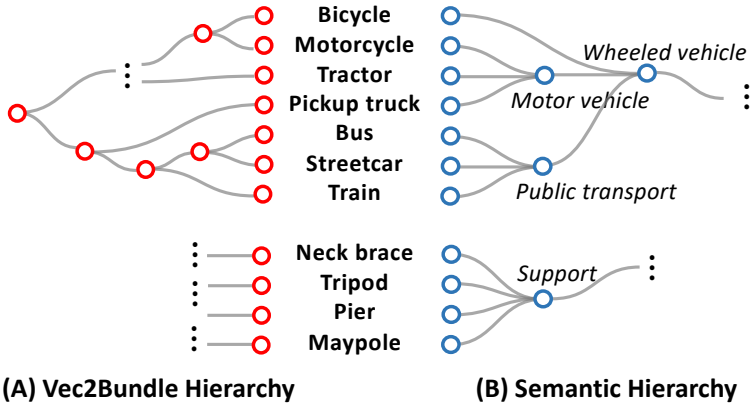


FIGURE 5.1: **Vec2Bundle learns a class hierarchy** that is fine-grained, captures visual confusion, and even carries certain semantics while being able to balance accuracy and specificity like traditional semantic hierarchies.

the intermediate semantic interpretations may very well be irrelevant. Furthermore, the way of grouping may not even be optimal due to bias in expertise or annotation collection. For instance, although the supernodes "bird" and "fish" are of the same depth in ImageNet, the sub-hierarchy of "bird" is much deeper than the sub-hierarchy of "fish". All in all, relying on semantic hierarchies to balance the accuracy and specificity trade-off is not always optimal.

In this chapter, we propose hierarchies learned directly from the data to optimize the accuracy *vs*. specificity trade-off. We argue that the main point behind a hierarchy should be to maximize the information gain, that is to minimize the number of misclassifications while maximizing the relevance of the prediction. As the misclassifications typically happen between visually similar classes, *e.g*. a "bus" and a "streetcar", by grouping the two we would arrive at a correct prediction while losing a bit of prediction relevance. Grouping classes has been proposed before [25, 5, 103, 129], using however existing semantic hierarchies like WordNet [112], iNaturalist [154], and Wikidata [155]. We propose, instead, to learn these hierarchies from the ground up and by combining classes that are confused together. Interestingly, this comes in contrast to standard discriminative learning that pushes the confused classes apart in order to maximize separation.

In this chapter, we propose a generalized accuracy-specificity trade-off framework that can tackle the above limitations. Our contribution can be summarized as follows: 1) We propose the Vec2Bundle hierarchy, a visual-data driven hierarchy that is learned through class prototype embedding, to help balance accuracy *vs*. specificity. 2) We propose a negation rule to compute a probabilistic hierarchy, making it possible to extend the current framework in balancing accuracy *vs*. specificity to multi-label image classification.

5.2 Related work

Hierarchical classification considers an image classification problem where a class hierarchy present. A class hierarchy H describes the abstractions among classes $\mathcal{V} = \{V_i\}_{i=1}^{M}$, which can be encoded as a set of acyclic "is-a" relations $\mathcal{E}=\{V_{\text{parent}} \leftarrow V_{\text{child}}\}$ [112]. The class hierarchy is often thought of as a tree [129, 118, 103] or in a more general way as a directed acyclic graph [26]. That is, $H = (\mathcal{E}, \mathcal{V})$. The task of hierarchical classification is to learn a classification function $f(\cdot)$ that maps the input image X to the output Y, which is possibly a node on the hierarchy [141]. In a probabilistic framework, the classifier function $f : X \mapsto Y$ can be decomposed into a *likelihood function* $\rho : X \mapsto P$, where $P=[p_1, \cdots, p_M]$ is the posterior for each class, and a *predictor function* $\pi : P \mapsto Y$ that decides the final output label(s) Y given the posteriors. That is $f(X) = (\pi \circ \rho)(X) \rightarrow Y$. Previously, extensive attention has been put on improving ρ for better top-1 accuracy and efficiency [129, 62, 172], making better mistakes [7], semantically aware in hyper-spherical space [43, 5] or hyperbolic space [77, 100, 103]. However, the attention on the predictor π has been largely overlooked. In this chapter, we mainly focus on the design of π to balance accuracy *vs*. specificity.

Accuracy-specificity trade-off can be achieved by a predictor π that follows a certain prediction policy. Depending on whether the structure between class labels are considered or not, we identify two types of predictors from the literature: unstructured and structured predictors. *Unstructured predictors* are a class of predictors that return a bundle of labels without considering the structure between class labels. In the one-*vs*.-rest classification, Top-K predictor is a natural fit [134, 83, 143, 75, 157, 91] where the trade-off between accuracy and specificity can be achieved by varying K, *i.e*. the trade-off parameter. As an example, Top-1 predictor $\pi=\arg\max(P)$ is the most specific, while in contrast, Top-5 predictor $\pi=\arg\text{sort5}(P)$ is more accurate. For multi-label classification threshold-based predictor is used [19, 29] to return a set of labels with confidences higher than a threshold, *i.e*. $\pi=\{l_k \mid p_k > \text{threshold}\}$. Li *et al*. [91] learns a linear regression model to predict the number of output labels K as well as a confidence threshold for each test image. Since this class of predictors does not take into account any prior knowledge of class-correlation, their prediction bundle consists of arbitrary combinations and in principle would have 2^N possible patterns (where N is the number of classes). This huge output space makes it difficult to develop an optimized trade-off between accuracy and specificity.

In contrast, a *structured predictor* makes predictions following fixed patterns. This is usually achieved by leveraging a class hierarchy. We consider the leaf nodes of the class hierarchy are the N most specific classes, *i.e*. $\mathcal{C}=[l_1, l_2 \cdots, l_N] \subset \mathcal{V}$. Given the class hierarchy, a predictor should predict

either leaf nodes (the most specific classes) or internal nodes. While model confidence increases from leaves to the root, the specificity of the prediction decreases since returning an internal node as prediction is essentially equivalent to making a prediction on the bundle of its leaf nodes. As a result, there are different predictors proposed in the literature to optimize such accuracy-specificity cost of selecting a node in the hierarchy [26, 67]. The MAX-REW predictor [26], $\pi_{\text{MAX-REW}}$, takes a confidence threshold θ as the trade-off parameter and predicts the node with the highest information gain among those with probabilities greater than or equal to θ. DARTS [26], π_{DARTS}, outputs the node with the maximum expected reward, which is learned as a function of a trade-off parameter λ. Since the class hierarchy regularizes the patterns of the prediction bundles, the output space for structured predictors is much smaller, *i.e.* $|\mathcal{V}|$. In this chapter, we continue the line of work on structured predictors. However, the hierarchy is to be learned from the data rather than predefined by human experts.

Building class hierarchy Hinton *et al*. [62] consider a classification problem on very big datasets with tens of thousands of classes. They build a two-layer hierarchy of classes by performing K-means clustering on the covariance matrix between classes. With the top-level classification problem being solved by a classifier called generalist, the second level classes on different branches are handled by a set of paralleled trained classifiers called specialists. Yan *et al*. [172] build a hierarchical deep CNN that follows a similar two-level hierarchy. Such hierarchy is the result of spectrum clustering on the confusion matrix from a flat classifier. Their model separates easy classes using a coarse category classifier while distinguishing difficult classes using fine category classifiers. Unlike [62], their model can be trained end-to-end. Bengio *et al*. [6] emphasize the importance of hierarchical structure in dealing with a large number of classes. They learn a disjoint tree-structure of labels by optimizing the overall tree loss that encourages the learnability of classification problems specified by such a tree. Following this direction, Deng *et al*. [25] further builds a balanced label tree to help accuracy *vs*. computational efficiency trade-off. They highlight that such a balanced label-tree can be simultaneously learned along with the set of classifiers associated with each tree node. In this chapter, we are more interested in building a class hierarchy that is dedicated to resolving confusion and thus favors the accuracy-specificity trade-off.

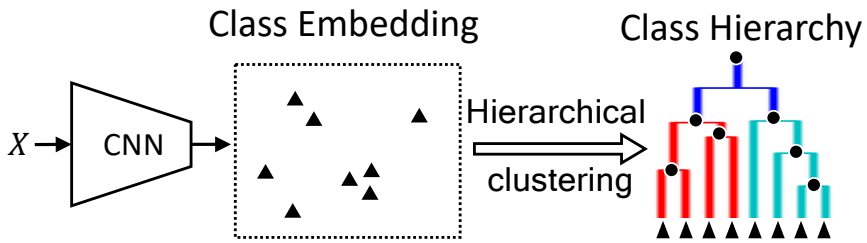


FIGURE 5.2: **Learning a class hierarchy** is achieved by first training a deep neural network to embed class-prototypes, and then applying a hierarchical clustering. We call this the Vec2Bundle hierarchy. In comparison to a semantic hierarchy, our Vec2Bundle hierarchy captures both the visual and semantic correlation between classes.

5.3 Learning Class Hierarchies

Although a semantic hierarchy ensures each node to have semantic meaning, it is often not the optimal structure when performing an accuracy *vs*. specificity trade-off. First, handcrafted hierarchies are not balanced due to human interests or expertise, and second, it does not take into account visual confusion. We propose a method to discover a hierarchy from the data. While, in theory, the learned hierarchy does not represent semantics, in practice we observe that a good proportion of the learned nodes correspond to semantic themes. As summarized in Fig. 5.2, our method discovers a class hierarchy by clustering a class prototype embedding learned from a classification network.

Class prototype embedding. To build a class prototype embedding, we start from the design of a fully-connected (FC) layer. In the last FC layer, given the feature vector $\vec{\mathbf{x}}$ for image X, the N-way logit outputs $[z_1, \cdots, z_N]^T$ are obtained by comparing $\vec{\mathbf{x}}$ with all pairs of weights and bias tied to each class:

$$X \xrightarrow[\text{base network}]{H(\cdot)} \vec{\mathbf{x}} \odot \begin{bmatrix} (\vec{\mathbf{w}}_1, b_1) \\ (\vec{\mathbf{w}}_2, b_2) \\ \vdots \\ (\vec{\mathbf{w}}_N, b_N) \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}, \tag{5.1}$$

where $\{(\vec{\mathbf{w}}_k^T, b_k) \,|\, k{=}1 \cdots N\}$ are FC parameters. In most works, the last layer is implemented as a linear mapping, *i.e.* $z_k{=}\vec{\mathbf{w}}_k^T\vec{\mathbf{x}} + b_k$, where the output logit z_k connects to the loss function. During training, $\vec{\mathbf{w}}_k$ and $\vec{\mathbf{x}}$ are updated simultaneously. We notice such a learning process does not yield a meaningful embedding space where $\vec{\mathbf{w}}_k$ and $\vec{\mathbf{x}}$ are comparable. This is because logit z_k does not explicitly encode the similarity between $\vec{\mathbf{w}}_k$ and $\vec{\mathbf{x}}$, but rather the relative strength w.r.t. other logits z_j ($j \neq k$). Thus, upon convergence, the value of $z_k \in (-\infty, +\infty)$ cannot interpret the correlation between $\vec{\mathbf{w}}_k$ and $\vec{\mathbf{x}}$.

As a consequence, the learned $\vec{\mathbf{w}}_k$ cannot be treated as an embedding of class prototypes.

To fix this, we propose to learn class prototypes by replacing the last linear mapping with a squared Euclidean distance mapping *i.e.*:

$$d_k = \frac{||\vec{\mathbf{w}}_k - \vec{\mathbf{x}}||^2}{\gamma_k}, \tag{5.2}$$

where γ_k is a positive scaling factor and $d_k \geq 0$. Given the squared Euclidean distances for all classes, *i.e.* $[d_1, \cdots, d_N]$, the final likelihood estimation of an image being the k-th class for a one-*vs*.-rest classification problem is computed together with softmax, which gives

$$p_k = \frac{e^{-d_k}}{\sum_i e^{-d_i}} = \frac{\exp\left(-\frac{||\vec{\mathbf{w}}_k - \vec{\mathbf{x}}||^2}{\gamma_k}\right)}{\sum_i \exp\left(-\frac{||\vec{\mathbf{w}}_i - \vec{\mathbf{x}}||^2}{\gamma_i}\right)} = \frac{\mathcal{K}(\vec{\mathbf{w}}_k, \vec{\mathbf{x}})}{\sum_i \mathcal{K}(\vec{\mathbf{w}}_i, \vec{\mathbf{x}})}, \tag{5.3}$$

where $\mathcal{K}(\cdot, \cdot) \in (0, 1]$ is an RBF kernel, a proper similarity measure between $\vec{\mathbf{w}}_k$ and $\vec{\mathbf{x}}$. In this setup, $\mathcal{K}(\vec{\mathbf{w}}_k, \vec{\mathbf{x}}) \to 1$ indicates an image X is closely correlated to the k-th class, and vice versa. Thus, once the majority of images from the same class are mapped to a shared local embedding space, $\vec{\mathbf{w}}_k$ naturally becomes their representative, *i.e.* the class-prototype. Moreover, when images from the j-th class and the k-th class have confusion with each other, both $\mathcal{K}(\vec{\mathbf{w}}_j, \vec{\mathbf{x}})$ and $\mathcal{K}(\vec{\mathbf{w}}_k, \vec{\mathbf{x}})$ will have a similarly high value, which helps to pull class-prototypes $\vec{\mathbf{w}}_j$ and $\vec{\mathbf{w}}_k$ close to each other. Subsequently, structures between classes are implicitly created. We refer to this proposed layer as the RBF embedding layer.

The likelihood estimation for *multi-label* classification problems is normally done by modeling N independent binary classifications using a sigmoid function. However, the sigmoid function requires an input value in $(-\infty, +\infty)$ whereas Eq. 5.2 only produces $d \in [0, +\infty)$. To bypass this limitation, instead of generating one single logit value per class, we propose to compute a pair of logits:

$$d_k^+ = \frac{||\vec{\mathbf{w}}_k^+ - \vec{\mathbf{x}}||^2}{\gamma_k} \quad \text{and} \quad d_k^- = \frac{||\vec{\mathbf{w}}_k^- - \vec{\mathbf{x}}||^2}{\gamma_k}, \tag{5.4}$$

where $\vec{\mathbf{w}}_k^+$ is the prototype for the k-th class and $\vec{\mathbf{w}}_k^-$ is an auxiliary prototype for not being the k-th class. The final likelihood estimation of p_k becomes a softmax form:

$$p_k = \frac{e^{-d_k^+}}{e^{-d_i^+} + e^{-d_i^-}} = \frac{\mathcal{K}(\vec{\mathbf{w}}_k^+, \vec{\mathbf{x}})}{\mathcal{K}(\vec{\mathbf{w}}_k^+, \vec{\mathbf{x}}) + \mathcal{K}(\vec{\mathbf{w}}_k^-, \vec{\mathbf{x}})}. \tag{5.5}$$

That is, the multi-label classification problem is solved by N independent binary softmax classifiers.

We learned the proposed embedding layer following standard Maximum Likelihood Estimation. Once the network converges, we extract the weights of the last embedding layer $\{\vec{\mathbf{w}}_k\}$ and treat them as class prototypes.

Extract class hierarchy. Given the learned class-prototypes $\{\vec{\mathbf{w}}_k\}$, next, we extract class hierarchy based on their distance in the embedding space. To this end, we compute the pairwise distance between the class-prototypes and performance an agglomerative hierarchical clustering [109] to group them into bundles in a bottom-up fashion. This results in a class hierarchy with its leaf nodes as all N classes. Since we learn class-prototype with a squared Euclidean space, we wish the linkage criteria, *i.e.* distance measure between clusters, also works in such space. To this end, we use Ward linkage criteria [164], where a pair of clusters (A, B) will be merged if the total within-cluster variance $d(A, B)$ is minimum, *i.e.*

$$d(A,B) = \sum_{i \in A \cup B} ||\vec{\mathbf{w}}_i - \vec{\mathbf{m}}_{A \cup B}||^2 - \sum_{i \in A} ||\vec{\mathbf{w}}_i - \vec{\mathbf{m}}_A||^2 - \sum_{i \in B} ||\vec{\mathbf{w}}_i - \vec{\mathbf{m}}_B||^2, \tag{5.6}$$

where $\vec{\mathbf{m}}_j$ is the center of cluster j. In doing so, we are able to discover the most confused combination of classes at each step, with the least loss in specificity. We coin such extracted hierarchy the Vec2Bundle hierarchy.

Now, with the learned Vec2Bundle hierarchy, a structured predictor is able to balance accuracy-specificity for one-*vs*.-rest classification, as it is empirically shown in our experiments. However, extending such a framework to multi-label classification is non-trivial as we will discuss next.

5.4 Extending to Multi-label Classification

Given a class hierarchy in multi-label classification, be it a Vec2Bundle hierarchy or a semantic hierarchy, we explore in this section for the first time how to perform the accuracy *vs*. specificity trade-off in a multi-label setting. In a probabilistic framework, the first step is to assign each node in the hierarchy with a probability score that is non-decreasing along the path to root. There are two approaches to achieve this: top-down [113, 129] and bottom-up [7].

The *top-down* approach [113, 129] follows the product rule [137, 79] and starts from the root to build a tree of probabilistic classifiers at each branching node. The probability for each node is conditioned on all the nodes from the root to that node. The *bottom-up* approach [7] follows the sum rule [183]. The probability of an internal node is computed by summing up the probability of its children. However, both approaches assume that the leaf nodes are

mutually exclusive. These assumptions are typically violated for multi-label classification. To this end, we propose a *negation rule*, for computing internal confidences where leaves are not mutually exclusive like in a multi-label scenario.

Negation rule. We start by revisiting the event of an image is eligible to be labeled with an internal node V. Assuming V has leaf nodes $\{l_1, \cdots, l_n\}$, such event is equivalent to an image has at least one label from the leaf nodes of V. Thus the event $\overline{V}$ happens when an image does not have any label from its leaf nodes. The probability can be written out as $P(\overline{V}) = P(\overline{l_1 \cdots l_n})$, which can be further expanded to $P(\overline{l_1}) \cdots P(\overline{l_n})$ by assuming that the leaf nodes are independent. Thus the probability for V can be written as:

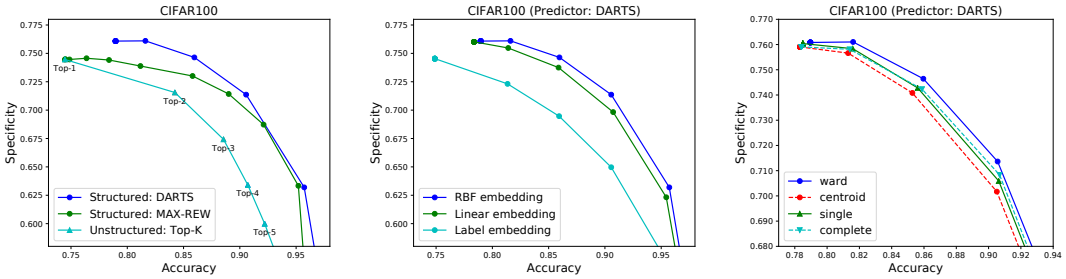
$$\begin{aligned} P(V) &= 1 - P(\overline{V}) = 1 - P(\overline{l_1}) \cdots P(\overline{l_n}) \\ &= 1 - \prod_{i=1 \cdot n} (1 - P(l_i)). \end{aligned} \tag{5.7}$$

It can be shown that the probabilistic hierarchy computed from the negation rule also satisfies the property of non-decreasing hierarchical probability, just as the product rule and sum rule do. That is, $P(A) \geq P(B)$ if node A is the ancestor of node B in the hierarchy. We show the proof in the supplementary materials. However, the negation rule does not ensure that the probability for the root node is 1. Therefore, we propose to re-calibrate the confidences by scaling the probability of all nodes by a factor of the reciprocal of the root probability, so that the root probability is always 1.

Predictor. Now, with predicting internal probabilities, a structured predictor is able to balance the accuracy-specificity trade-off. We use a variant of the MAX-REW predictor, called Max$K$-REW, for multi-label classification. Rather than returning a single prediction that maximizes the reward in specificity, a Max$K$-REW returns K most specific predictions on the hierarchy among those nodes satisfying a confidence threshold, *i.e.* trade-off parameter, θ. Thus, varying θ from low to high results in multi-label predictors that range from the most specific to the most accurate.

5.5 Experiments

In this section, we evaluate the accuracy *vs*. specificity trade-off with our Vec2Bundle hierarchies for general classification problems on several datasets. We first detail our experimental setup in Seciton 5.5.1. Then, we provide an ablation study in Section 5.5.2. It is followed by the evaluation of trade-off performance for one-*vs*.-rest classification in Section 5.5.3. Last, we demonstrate a multi-label trade-off classification in Section 5.5.5.



(A) **Structured or unstructured predictor?** (B) **Influence of embeddings** (C) **Sensitivity to linkage criteria**

FIGURE 5.3: **Ablations** on Vec2Bundle hierarchy for single-label image classification. Equipping structured predictors with class hierarchy prevails over unstructured predictors. In building such a class hierarchy, the RBF class-prototype embedding and ward linkage criteria for class clustering are essential.

5.5.1 Experimental setup

Datasets. Although the proposed Vec2Bundle hierarchy is motivated by the absence of semantic hierarchy in the majority of classification, for sake of comparison, we experiment with datasets that include a semantic hierarchy. Specifically, we consider ***CIFAR100*** [82], ***ILSVRC65*** [26] and ImageNet Large Scale Visual Recognition (***ILSVRC***) 2012 [27] for one-*vs*.-rest classification problem, and ***MS-COCO14*** [97] for multi-label classification. These 4 datasets have 100, 57, 1000, 80 classes respectively, which form the leaf nodes of the hierarchy.

Hierarchies. The semantic hierarchy of CIFAR100 is created by Barz and Denzler [5], which is a strict tree-shaped structure. The hierarchy has 62 internal nodes in total, with a maximum depth of 8. Following the WordNet [112] ontology, Deng *et al*. [27] construct ImageNet by collecting and organizing images into a semantic hierarchy. The original hierarchy is not a strict tree. Following [129], we use a trimmed version of the hierarchy that keeps the shortest path to the root for each node if there is more than one path. The hierarchy has 372 internal nodes, with a maximum depth of 16. ILSVRC65 has the simplest semantic hierarchy with 8 internal nodes with a maximum depth of 16. For MS-COCO, we adapt the semantic hierarchy for the "things" category of COCO-Stuff [13]. We clean the original hierarchy by renaming the duplicate node names with different names. All hierarchies will be released.

Implementation. Since the likelihood function is not the focus of this paper, we always adopt ResNet18 [57] as the backbone for assigning class confidences to each image. All the models are trained from scratch with the weights of the backbone network being randomly initialized following He *et*

al. [58]. For our proposed FC layer, we randomly initialize weights $\vec{\mathbf{w}}$ from a Gaussian distribution $N(0, 0.005)$ and fix γ as constant 1. The whole network is trained with an SGD optimizer with momentum set to 1e-4. We use a weight decay of 0.0005 for CIFAR100 and 0.0001 for the other datasets. Note that we exclude the learnable weights of the last layer from being penalized by the weight decay regularization. This gives the class prototypes more freedom to move in the embedding space so that class-to-class relationships can be captured. We train the network for 120 epochs on CIFAR100, and 90 epochs on the other datasets. The initial learning rate is set to 0.1 and decreased by a factor of 10 at 50% and 75%. As predictor, we use DARTS and MAX-REW with the trade-off parameter $\theta{=}[0, 0.1, \cdots, 1.0]$. Since the DARTS predictor needs a hold-out dataset to learn the trade-off parameter, we always split the original training set into a train-set and validation-set with a ratio of 4:1. The evaluation is always carried out on the original test set.

Metrics Following [26], we use a normalized version of information gain to measure the specificity of a prediction. Given a prediction bundle $Y{=}\{l_i \mid i \in 1 \cdots N\}$, the specificity is defined as:

$$r(Y) = 1 - \frac{\log_2 |Y|}{\log_2 N}, \tag{5.8}$$

where N is the total number of classes and $|Y|$ is the size of the prediction bundle. Therefore, in the case of a structured predictor on the class hierarchy, making predictions on a single leaf node receives a specificity 1 ($|Y|{=}1$) and at root node receive 0 ($|Y|{=}N$). Further, the accuracy and specificity for a predictor π on the entire test dataset is defined as the expectation over each test sample, *i.e.*:

$$Acc(\pi) = \mathrm{E}(\mathbb{1}\left[|Y \cap \hat{\mathcal{Y}}| > 0\right]), \tag{5.9}$$

$$Spec(\pi) = \mathrm{E}(\mathbb{1}\left[|Y \cap \hat{\mathcal{Y}}| > 0\right] \cdot r(Y)), \tag{5.10}$$

where $\hat{\mathcal{Y}}$ is the ground truth and $\mathbb{1}\left[\cdot\right]$ denotes the Iverson bracket that returns 1 when the expression inside is true, and 0 otherwise.

5.5.2 Ablations

We ablate Vec2Bundle's ability to balance accuracy and specificity in image classification on CIFAR100.

Structured or unstructured predictor? In this ablation, we compare the Top-K unstructured predictor with the structured MAX-REW and DARTS [26] predictors. The structured predictors use our Vec2Bundle hierarchy. We show the comparison in Fig. 5.3a. The MAX-REW predictor and Top-K predictor start at the same point on the upper left. This is because when the

trade-off parameter θ of MAX-REW starts at 0, MAX-REW predicts the most likely leaf node, which is the same behavior as a Top-1 predictor. As θ and K increases, we observe MAX-REW clearly outperforms Top-K. This demonstrates that the structured bundles defined by the class-hierarchy are more effective in resolving confusion than the unstructured bundles of a fixed size of K. Further, we observe DARTS obtains better performance than MAX-REW. This owns to DARTS ability to learn the optimal trade-off parameters on a validation set while MAX-REW used pre-defined trade-off parameters. We conclude class-hierarchy helps the structured predictor to regularize the predictor and outperform the unstructured predictor. In the following experiments, we only report results for the best performing structured predictor, *i.e.* DARTS.

Influence of embedding. Next, we verify the influence of the embedding when building the Vec2Bundle hierarchy, with the same clustering algorithm. We consider two other embeddings in addition to our class-prototype embedding: 1) the weights of the last layer implemented by the linear mapping and 2) a label embedding method by Habibian *et al*. [54] that maps class-prototypes and image representation into a shared space and optimizes with a quadratic error loss. We report results in Fig. 5.3b. We observe our RBF embedding obtains the best performance. The performance of the label embedding method is sub-optimal, we suspect this is because the quadratic error loss is not as stable as the cross-entropy loss where the RBF layer and linear layer are learned. We conclude the proposed class-prototype embedding is most suitable for building the Vec2Bundle hierarchy.

Sensitivity to linkage criteria Last, we show how different linkage criteria, *i.e.* the distance measure between two clusters, for the hierarchical clustering affect our result. We compare our choice, *i.e.* Ward's criteria, with three other common choices: 'centroid'-linkage, 'single'-linkage, and 'complete'-linkage. The 'centroid'-linkage considers the Euclidean distance between the centroid of each cluster, *i.e.* the arithmetic mean of all elements. In contrast, the 'single'-linkage and 'complete'-linkage consider the nearest or farthest pair of elements from two clusters respectively. We report results in Fig. 5.3c.

Despite the 'centroid'-linkage being the most natural choice, it results in a hierarchy that performs the worst for trade-off. We suspect this is because these criteria rely on Euclidean distance whereas the class-prototype is learned in the squared Euclidean distance space. Further, we observe 'complete'-linkage performs slightly better than 'single'-linkage, possibly due to the fact of avoiding 'chaining phenomenon'. The 'ward'-linkage performs the best, it is able to find the two most confused clusters to merge. We conclude the hierarchical clustering is sensitive to linkage criteria and it is crucial to take into account the learning space of the class-prototypes in choosing the proper linkage criteria. Thereupon, we stick to 'ward'-linkage in our experiments.

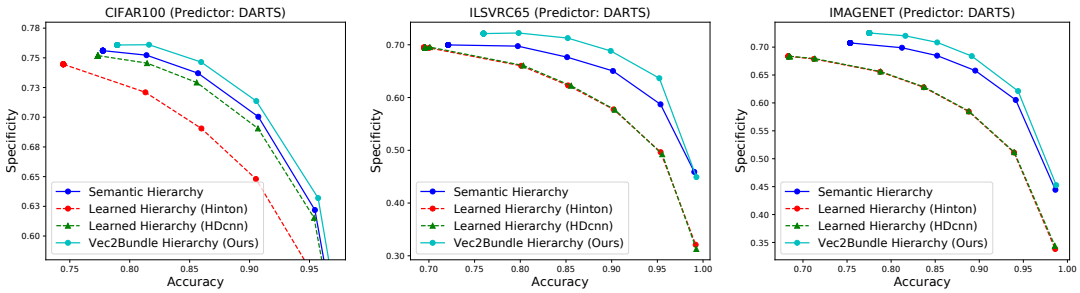


FIGURE 5.4: **Balancing accuracy *vs*. specificity for single-label image classification** by comparing Vec2Bundle with alternative hierarchies. While a semantic hierarchy demonstrates superiority over two previously learned hierarchies [62, 172], our proposed Vec2Bundle hierarchy balances accuracy and specificity best.

5.5.3 Balancing accuracy *vs*. specificity for single-label image classification

Setup. Next, we compare our Vec2Bundle hierarchy with alternative hierarchies in terms of their accuracy-specificity trade-off performance on three image classification datasets. As baselines, we consider two learned class hierarchy alternatives proposed by Hinton *et al*. [62] and Yan *et al*. [172]. Hinton *et al*. [62] generate their hierarchy by applying K-means clustering on the covariance matrix w.r.t. each class. Yan *et al*. [172] resort to spectral clustering on a confusion matrix of the validation set. For both of the two methods, we set the ratio of the number of clusters *vs*. the number of leaf nodes as 1:10. A summary of the hierarchy statistics is provided in the supplementary materials.

Result. We report results in Fig. 5.4. We first observe the HDcnn hierarchy by Yan *et al*. [172] demonstrates better performance on CIFAR100 than the one by Hinton *et al*. [62]. This is because, in HDcnn, the ground truth labels from the validation set are used to construct a confusion matrix, which captures more accurate model confusions. Hinton *et al*. [62] prefer a simpler solution with a covariance matrix where a validation set is not needed. Interestingly, we observe the two baseline methods have similar performance on ILSVRC65 and ImageNet. This is partially due to the image resolution of CIFAR100 (32x32) being much lower than ILSVRC65 and ImageNet (224x224). As a result, the visual confusion on ILSVRC65 and ImageNet is hard to be captured by simply clustering on the confusion/co-variance matrix. Hence, the semantic hierarchy performs better than the two baselines on all datasets, showing semantic groupings are more efficient in resolving confusion. Overall, we observe our Vec2Bundle hierarchy consistently outperforms all hierarchies, be they learned or semantic, on all datasets. We attribute this to

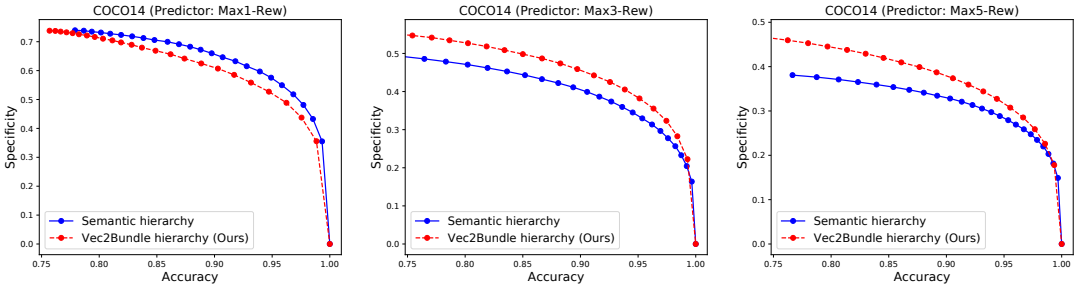


FIGURE 5.5: **Balancing accuracy *vs.* specificity for multi-label image classification** by comparing Vec2Bundle with a semantic hierarchy. Vec2Bundle hierarchy helps MaxK-REW predictor achieve better balancing performance in multiple predictions.

the learned class-prototype embedding and the agglomerative clustering in discovering the hierarchy. We conclude Vec2Bundle hierarchy is suitable for balancing accuracy and specificity in single-label image classification.

5.5.4 Semantics in the Vec2Bundle hierarchy

Setup. In this experiment, we assess whether the learned Vec2Bundle hierarchy captures semantics. Since the semantic hierarchy is the only source of semantics that we are certain about, we propose to match for each internal node $Y=\{l_i\}$ on a source semantic hierarchy H_{src} to a closest internal node $Z=\{l_j\}$ on the target Vec2Bundle hierarchy H_{tgt}. The quality of all such matchings reflect how much semantics the target Vec2Bundle hierarchy carries. Quantitatively, we measure this with an overall matching score defined as:

$$m(H_{\text{src}} \rightarrow H_{\text{tgt}}) = \mathrm{E}_{Y \in H_{\text{src}}} \left(\max_{Z \in H_{\text{tgt}}} \frac{|Y \cap Z|}{|Y \cup Z|} \right), \tag{5.11}$$

where $|Y \cap Z|$ and $|Y \cup Z|$ are the cardinalities of the intersection and the union of set Y and Z in term of the leaf nodes. As such, if every semantic node Y can be exactly matched to a Vec2Bundle node Z, the overall matching score will be 1. Note that this measure is non-symmetric, *i.e.* $m(H_{\text{src}} \rightarrow H_{\text{tgt}}) \neq m(H_{\text{tgt}} \rightarrow H_{\text{src}})$. We evaluate on CIFAR100 and ImageNet the matching score from the semantic hierarchy to other Vec2Bundle hierarchies learned by Hinton *et al*. [62], HDcnn [172] and ours. As references, we also report the matching score between the semantic hierarchy on CIFAR100 we used throughout the experiment [5] and another simpler semantic hierarchy $H^{(\text{simple})}_{\text{semantic}}$ defined by Krizhevsky *et al*. [82] that only contains two levels of coarse categories and fine-grained categories.

Source → Target	CIFAR100	ImageNet
$H_{\text{semantic}} \rightarrow H_{\text{vec2bundle}}$ (Hinton)	0.47	0.31
$H_{\text{semantic}} \rightarrow H_{\text{vec2bundle}}$ (HDcnn)	0.61	0.32
$H_{\text{semantic}} \rightarrow H_{\text{vec2bundle}}$ (Ours)	**0.74**	**0.56**
$H_{\text{semantic}} \rightarrow H^{(\text{simple})}_{\text{semantic}}$	*0.70*	-
$H^{(\text{simple})}_{\text{semantic}} \rightarrow H_{\text{semantic}}$	*0.78*	-

TABLE 5.1: **Semantics in the Vec2Bundle hierarchy**. Our Vec2Bundle hierarchy performs the best among the competitors, and comes close to the reference score between two semantic hierarchies, showing that a good amount of semantics is captured by our Vec2Bundle hierarchy without being explicitly instructed to do so.

Result. We report results in Table 5.1. Hierarchies from HDcnn show better semantic matching than hierarchies from Hinton *et al*. We attribute this to the extra information of the ground truth label on the validation set HDcnn uses. Overall, our Vec2Bundle hierarchy outperforms both HDcnn and Hinton *et al*. on both CIFAR100 and ImageNet, with a matching score of 0.74 and 0.56 respectively. In the meanwhile, we notice that the matching score between two semantic hierarchies on CIFAR100 is 0.70 and 0.78, which are quite close to the matching score of ours, *i.e*. 0.74. This shows a good proportion of nodes on our Vec2Bundle hierarchy are semantically meaningful. Further, we show a 2D visualization of the class-prototype embedding with T-SNE [105] in supplementary materials, where the captured semantics can be identified. We conclude our Vec2Bundle hierarchies are able to learn a certain amount of semantics.

5.5.5 Balancing accuracy *vs*. specificity for multi-label image classification

Setup. Last, we evaluate the accuracy-specificity trade-off for multi-label classification on the MS-COCO14 dataset. We consider both the semantic hierarchy and Vec2Bundle hierarchy learned with Eq. 5.5. As mentioned in Section 5.4 the leaves are not mutually exclusive and the associated classifiers are independently trained, so we adopt the negation rule (Eq. 5.6) in deriving a probabilistic hierarchy for the trade-off. Note that directly extending the information gain to define specificity on multi-label classification can be nontrivial since the output follows a Multivariate Bernoulli distribution [23], which has a rather complex probability mass function. Thus, we simply

borrow the form of the specificity definition in Eq. 5.8 for each of the multi-label predictions, and the total specificity w.r.t. one test example is their average. Since the multi-label predictor π returns multiple predicted nodes on the hierarchy per image example, the accuracy definition is slightly different. Specifically, each prediction is evaluated to be 1 if at least one of the leaf nodes hit the ground labels, otherwise 0. We average all evaluation outcomes on one image to get correctness value per-image. The final accuracy is the expectation of the correctness value on all test images. We evaluate Max1-REW, Max3-REW, and Max5-REW predictor in this experiment.

Results. We report results in Fig. 5.5. First, all of the three plots show accuracy *vs*. specificity is possible for multi-label classification, justifying the correctness of the proposed negation rule in deriving a probabilistic hierarchy. Second, we observe that the semantic hierarchy has a better performance with the Max1-REW predictor. This is because a predictor with Vec2Bundle hierarchy tends to predict bundle with smaller sizes, resulting in a low recall of ground truth labels. However, when the number of predictions increases to 3 and 5, *i.e*. Max3-REW and Max5-REW, the proposed Vec2Bundle hierarchy consistently outperform semantic hierarchy. We attribute this to the captured relationships, such as visual similarity and contextual co-occurrence, between class by our embedding space. We show the 2D visualization of class-prototype embedding on MS-COCO14 in the supplementary materials. We conclude balancing accuracy *vs*. specificity on multi-label classification is feasible with class hierarchies, and the Vec2Bundle hierarchy helps MaxK-REW predictor achieve better balancing performance in multiple predictions.

5.6 Conclusion

We introduce a visual data-driven hierarchy, called Vec2Bundle, for balancing accuracy *vs*. specificity in image classification. The Vec2Bundle hierarchy is extracted from a class-prototype embedding layer that is trained discriminatively. In comparison to a semantic hierarchy, our Vec2Bundle hierarchy is cheap to obtain and shown to better balance accuracy *vs*. specificity. Surprisingly, we find the Vec2Bundle hierarchy carries a considerable amount of semantics without being instructed to do so. Further, we introduce a new tool, called the negation rule, in computing the probabilities on the hierarchy that has non-mutually exclusive leaf nodes. This enables Vec2Bundle to be extended to multi-label classification. We conclude Vec2Bundle hierarchy is suited to balance accuracy *vs*. specificity for both one-*vs*.-rest classification and multi-label classification.

Chapter 6

Conclusion

This thesis strives to answer "*How to better utilize geometry for better image understanding?*". This question is answered from two aspects, namely, geometry in visual image understanding and geometry in semantic image understanding. In Part I of this thesis, we study the roles of 3D geometry for visual image understanding. We first show that it is possible to automatically explain a variety of visual contents in the image with texture-free 3D shapes, by searching the closest 3D shape from a database and then matching it onto the 2D image. Furthermore, we develop a deep learning framework to reliably recover a set of 3D geometric attributes from a 2D image. In Part II, we explore label geometry for semantic image understanding. We show that a set of image classification problems have geometrically similar probability spaces. To this end, we introduce label geometry, unifying one-*vs*.-rest classification, multi-label classification, and out-of-distribution classification in one framework. Moreover, we show that we can learn hierarchical label geometries to better model image classification tasks, when a class hierarchy is used to balancing the accuracy and specificity of an image classifier. Now, we return to the four research questions raised in Chapter I.

6.1 Part I. Deep learning with 3D geometry

How to search and match texture-free 3D shapes to a 2D image?

We answer this question in Chapter 2, where we set our goal as searching and matching the best-rendered view of a texture-free 3D shape to an object of interest in a 2D query image. Matching rendered views of 3D shapes to RGB images is challenging because, 1) 3D shapes are not always a perfect match for the image queries, 2) there is great domain difference between rendered and RGB images, and 3) estimating the object scale versus distance is inherently ambiguous in images from uncalibrated cameras. In this chapter, we propose a deeply learned matching function that attacks these challenges and can be used for a search engine that finds the appropriate 3D shape and matches it to objects in 2D query images. We evaluate the proposed matching function and search engine with a series of controlled experiments on the 24

most populated vehicle categories in PASCAL3D+. We test the capability of the learned matching function in transferring to unseen 3D shapes and study overall search engine sensitivity w.r.t. available 3D shapes and object localization accuracy, showing promising results in retrieving 3D shapes given 2D image queries.

While the deeply learned matching function shows its viability in comparing the rendered view to a 2D image, the drawback of this approach is its computational cost - a total of 2,541 comparisons are encountered in processing a single image. As a future work, one direction to reduce the number of comparisons would be to integrate the viewpoint estimation from a single image. In this case, we may generate a quick initial guess of the 3D object pose from viewpoint estimation. Then, comparisons between rendered views and a 2D image are restricted to a few nearby poses.

Is it possible to train deep neural networks that output continuous estimations of viewpoints, rotations, and surface normal from a 2D image in a reliable and accurate manner?

This research question is answered in Chapter 3. We depart from the observation that many computer vision challenges require continuous outputs, but tend to be solved by discrete classification. The reason is the natural containment of classification is within a probability n-simplex, as defined by the popular softmax activation function. Regular regression lacks such a closed geometry, leading to unstable training and convergence to suboptimal local minima. Starting from this insight we revisit regression in convolutional neural networks. We observe that many continuous output problems in computer vision are naturally contained in closed geometrical manifolds, like the Euler angles in viewpoint estimation or the normals in surface normal estimation. A natural framework for posing such continuous output problems are n-spheres, which are naturally closed geometric manifolds defined in the $\mathbb{R}^{(n+1)}$ space. By introducing a spherical exponential mapping on n-spheres at the regression output, we obtain well-behaved gradients, leading to stable training. We show how our spherical regression can be utilized for several computer vision challenges, specifically viewpoint estimation, surface normal estimation, and 3D rotation estimation. For all these problems our experiments demonstrate the benefit of spherical regression.

One limitation of the proposed spherical regression is that it can only handle the spherical 3D targets. However, we may also be interested in quite a few 3D targets that are non-spherical, for example, depth information and 3D bounding box. These targets are quite common in 3D computer vision and are mostly linear. Thus, for further work, it is worthwhile to extend the spherical regression for the linear targets. For this purpose, the challenge could be to find a natural containment in linear space in replacement of a n-sphere.

6.2 Part II. Deep learning with label geometry

How to leverage the label geometry to unify image classifiers?

This question is answered in Chapter 4. In classical image classification problems, the softmax and binary classifier are commonly preferred solutions. We note that softmax has a closed label geometry, *i.e*. the probability simplex, and thus can be trained stably. However, as softmax is specifically designed for categorical classification, it assumes each image has just one class label. This limits its applicability for problems where the number of labels does not equal one, most notably zero- and multi-label problems. In these challenging settings, binary classifiers are, in theory, better suited. However, as they ignore the correlation between classes, they are not as accurate and scalable in practice. In this chapter, we start from the observation that the only difference between binary and softmax classifiers is their normalization function. Specifically, while the binary classifier self-normalizes its score, the softmax classifier combines the scores from all classes before normalization. Based on this observation, we introduce a normalization function that is learnable, constant, and shared between classes and data points. By doing so, we arrive at a new type of binary classifier that we coin quasibinary classifier. We show in a variety of image classification settings, and on several datasets, that quasibinary classifiers are considerably better in classification settings where regular binary and softmax classifiers suffer, including zero-label and multi-label classification. What is more, we show that quasibinary classifiers yield well-calibrated probabilities allowing for direct and reliable comparisons, not only between classes but also between data points.

Despite quasibinary classifiers do not need to know the number of labels per sample at test time, they largely rely on knowing the accurate number of labels at training time. However, previous works [171, 1, 180, 123, 175] show that the annotations we use to train classification models are in fact noisy - it is not rare to have missing labels or wrong labels for training samples. This may cause quasibinary classifiers to have biased estimations of normalization function at training time. For this reason, we believe studying the normalization function in the noisy annotation setting is a worthwhile further direction for quasibinary classifiers to pursue.

How to infer the label geometry from image classification to balance accuracy* vs. *specificity?

This question is answered in Chapter 5. In image classification, a class hierarchy is an import label geometry to balance accuracy *vs*. specificity. With the class hierarchy, a classifier may choose to predict an internal node instead of the most specific leaf node to avoid confusion, and by doing so,

improve accuracy. Previous approaches in balancing accuracy *vs*. specificity rely on manually defining a semantic hierarchy of classes, and are also limited to the one-*vs*-rest classification problems. In this chapter, we introduce a new embedding layer that is able to learn by discriminative training of class-prototypes. From the trained class-prototypes then a visual hierarchy is extracted. We refer to this embedding as *Vec2Bundle*. Further, by introducing a negation rule in deriving probabilities on the hierarchy, we are the first to enable a trade-off on multi-label hierarchical classification, wherein previous approaches that was infeasible due to the exclusiveness of leaf nodes. We validate the effectiveness of Vec2Bundle key components with ablation experiments and compare with the state-of-the-art in balancing accuracy and specificity for both single-label and multi-label image classification. Interestingly, it appears that Vec2Bundle can capture semantics without being explicitly instructed to do so.

Setting aside we have shown the learned hierarchies are better at balancing accuracy and specificity than manually defined counterparts, the learning stage of a hierarchy and the balancing stage for accuracy and specificity are independently optimized in our solution. Thus, finding an end-to-end solution that directly optimizes a class hierarchy towards an optimal performance in balancing accuracy *vs*. specificity can be a fruitful direction to explore in the future.

6.3 Closing remarks

In this thesis, we have explored the geometry for image understanding. Geometry, as a structural descriptor, exists ubiquitously not only in 3D space but also in semantic space. While we are only able to skim the surfaces of both topics through a limited stretch of discussions, we have revisited the importance of 3D geometry for visual image understanding and have generalized the label geometry in probabilistic models for semantic image understanding. Under the current frameworks of deep learning, we have shown geometry is either useful in explaining the visual content of a 2D image or is crucial in building probabilistic models to associate images with semantics. Along the journey in making computers to see the world in the same way as we human beings see, we believe endeavors in exploiting the 3D geometry and label geometry are valuable and rewarding.

Appendix A

Supplementary Materials for Spherical Regression

A.1 S^1: Viewpoint estimation with Euler angles

We show the viewpoint estimation network architecture used in this chapter in Fig. A.1. Given ResNet101 as backbone to provide a shared Pool5 feature (with 2048 output unit), we have 3 branches to estimate azimuth, elevation and in-plane rotation (theta) angels. Each branch begins with a fully-connected layer (Fc8), with 1024 output units, and makes a prediction for the 12 categories in Pascal3D+. Our prediction head is composed of two components: 1) absolute value prediction and 2) sign prediction.

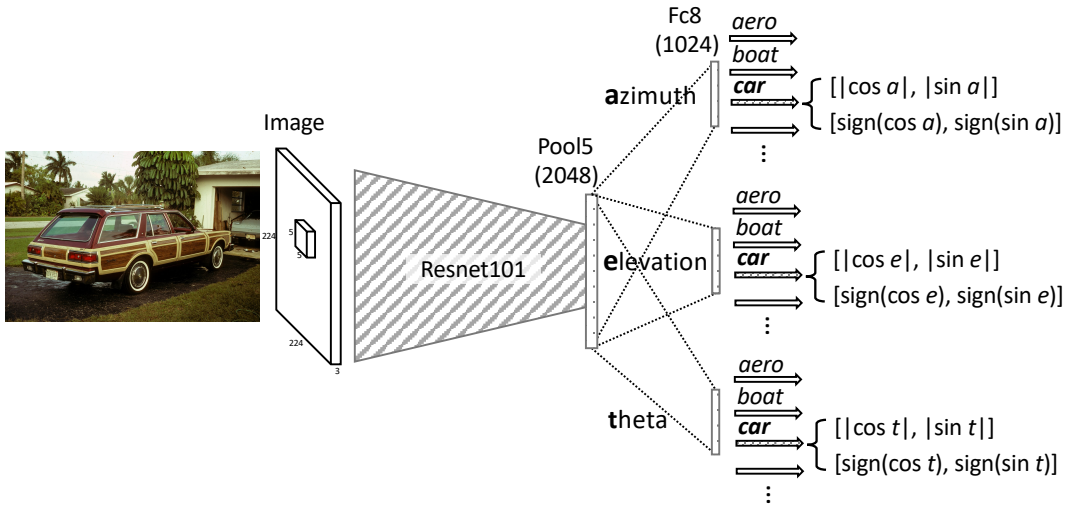


FIGURE A.1: **Network architecture for viewpoint estimation by Euler angles on Pascal3D+.**

We show the fine-grained evaluation in Table. A.1. In comparison with Penedones *et al*. [124], spherical regression improves the performance in all evaluation metrics, namely Acc@$\{\frac{\pi}{6}, \frac{\pi}{12}, \frac{\pi}{24}\}$.

We report the class-wise performance comparison in Table. A.2. Prokudin *et al*. [125] wins the most categories under MedError metric (5 out of 12). However, they made a larger mistake on difficult categories like boat, where the visual appearance has larger variance. For Acc@$\frac{\pi}{6}$ metric, our method wins the most (6 out of 12 categories). In comparison with Penedones *et al*. [124], adding spherical regression module consistently helps increase the accuracy across almost all categories.

TABLE A.1: **Viewpoint estimation with fine-grained evaluation on Pascal3D+**. We report results of **Acc@**$\{\frac{\pi}{6}, \frac{\pi}{12}, \frac{\pi}{24}\}$ ↑. Results generated by spherical regression module (S^3_{exp}) have a better alignment to the ground truth models.

	MedErr↓	**Acc@**$\frac{\pi}{6}$ ↑	**Acc@**$\frac{\pi}{12}$ ↑	**Acc@**$\frac{\pi}{24}$ ↑
Penedones *et al*. [124]†	11.6	83.6	66.3	35.9
This chapter: [124]†+ S^1_{exp}	**9.2**	**88.2**	**74.1**	**46.0**

† Based on our implementation.

TABLE A.2: **Category-wise evaluation of viewpoint estimation on Pascal3D+**.

	Method	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	mean
MedError	Mahendran *et al*. [107]	14.5	22.6	35.8	9.3	4.3	8.1	19.1	30.6	18.8	13.2	7.3	16.0	16.6
	Tulsiani *et al*. [151]	13.8	17.7	21.3	12.9	5.8	9.1	14.8	15.2	14.7	13.7	8.7	15.4	13.6
	Mousavian *et al*. [114]	13.6	12.5	22.8	8.3	3.1	5.8	11.9	12.5	12.3	12.8	6.3	11.9	11.1
	Su *et al*. [145]	15.4	14.8	25.6	9.3	3.6	6.0	9.7	10.8	16.7	9.5	6.1	12.6	11.7
	Penedones *et al*. [124]†	12.3	**11.5**	31.3	6.9	4.4	7.1	12.2	13.9	13.1	**7.7**	7.0	12.1	11.6
	Prokudin *et al*. [125]	9.7	15.5	45.6	**5.4**	**2.9**	**4.5**	13.1	12.6	**11.8**	9.1	**4.3**	12.0	12.2
	Grabner *et al*. [50]	10.0	15.6	**19.1**	8.6	3.3	5.1	13.7	11.8	12.2	13.5	6.7	11.0	10.9
	Mahendran *et al*. [108]	**8.5**	14.8	20.5	7.0	3.1	5.1	9.3	11.3	14.2	10.2	5.6	11.7	10.1
	This chapter: [124]†+ S^1_{exp}	9.2	11.6	20.6	7.3	3.4	4.8	**8.2**	**8.5**	12.1	8.7	6.1	**10.1**	**9.2**
$Acc@\pi/6$	Mahendran *et al*. [107]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Tulsiani *et al*. [151]	0.81	0.77	0.59	0.93	**0.98**	0.89	0.80	0.62	0.88	0.82	0.80	0.80	0.808
	Mousavian *et al*. [114]	0.78	0.83	0.57	0.93	0.94	0.90	0.80	0.68	0.86	0.82	0.82	0.85	0.810
	Su *et al*. [145]	0.74	0.83	0.52	0.91	0.91	0.88	0.86	0.73	0.78	0.90	**0.86**	0.92	0.820
	Penedones *et al*. [124]†	0.80	0.85	0.48	**0.96**	0.94	0.91	0.84	0.70	0.86	0.95	0.84	0.91	0.836
	Prokudin *et al*. [125]	**0.89**	0.83	0.46	**0.96**	0.93	0.90	0.80	**0.76**	0.90	0.90	0.82	0.91	0.838
	Grabner *et al*. [50]	0.83	0.82	**0.64**	0.95	0.97	0.94	0.80	0.71	0.88	0.87	0.80	0.86	0.839
	Mahendran *et al*. [108]	0.87	0.81	**0.64**	**0.96**	0.97	**0.95**	0.92	0.67	0.85	0.97	0.82	0.88	0.859
	This chapter: [124]†+ S^1_{exp}	0.88	**0.88**	0.61	**0.96**	0.97	0.93	**0.93**	0.74	**0.93**	**0.98**	0.84	**0.95**	**0.882**

† Based on our implementation.

A.2 S^2: Surface normal estimation

We show the visualization of surface normal prediction in Fig. A.2. The results from Zhang *et al*. [179] are smoother than our results from spherical

regression, but it makes some mistake with quite large surface area, *e.g.* the wall on the picture at row 3 column 2. In terms of boundaries, our results tend to be sharper. This is mainly due to the classification branch, which forces the prediction to choose the main direction in one out of four quadrants. Overall, our results maintain more details than Zhang *et al.* [179].

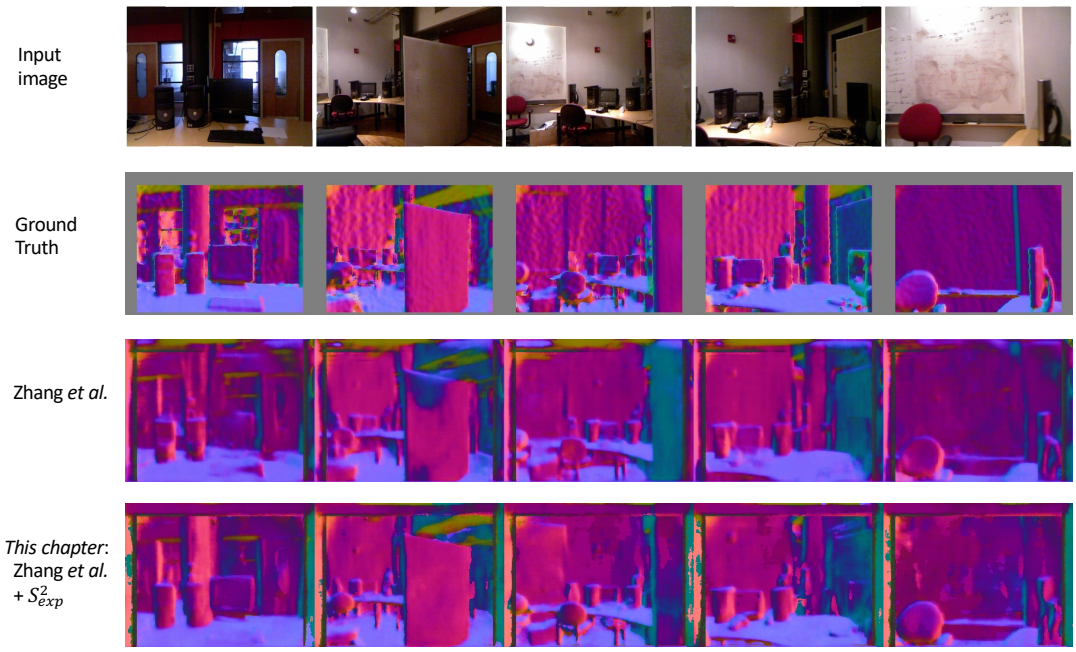


FIGURE A.2: **Visualization of Surface Normal Estimation on *NYU v2***. Predictions are made by model: "Zhang *et al.* [179]" and "Zhang *et al.* [179] + S^2_{exp}". While results from Zhang *et al.* [179] are smoother, our method generates sharp boundaries and thus maintains details.

A.3 S^3: 3D Rotation estimation with quaternions

We show a class-wise performance comparison based on Acc@$\frac{\pi}{6}$ in Fig. A.3. Since we are predicting the 3D rotation just from a single image, it can be seen that categories with high degree of symmetry have worse performance, *e.g.* bathtub, desk, night-stand and table. In comparison with the regression of quaternion with flat VGG16, spherical regression consistently helps increase the accuracy.

We show a visualization of 3D rotation estimation in Fig. A.4. The first row is the ground truth input images. We render the predicted rotations from VGG16 and VGG16+S^3_{exp} in second and third rows. We can see our result have a better alignment to the ground truth models.

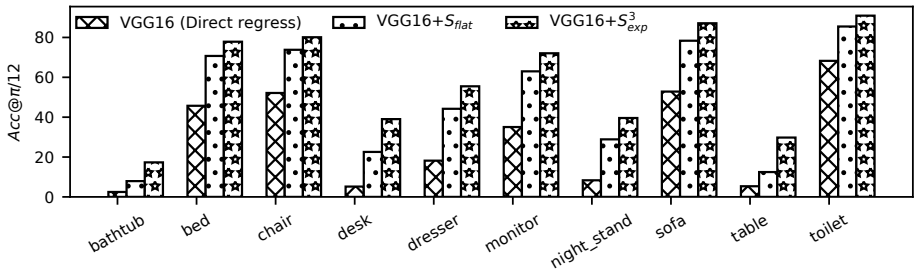


FIGURE A.3: **Class-wise comparison of 3D rotation estimation on *ModelNet10-SO3***. Categories with high degree of symmetry are observed to have worse performance, *e.g.* bathtub, desk, night-stand and table. Spherical regression module (S^3_{exp}) consistently helps increase the performance over flat regression of quaternion by VGG16.

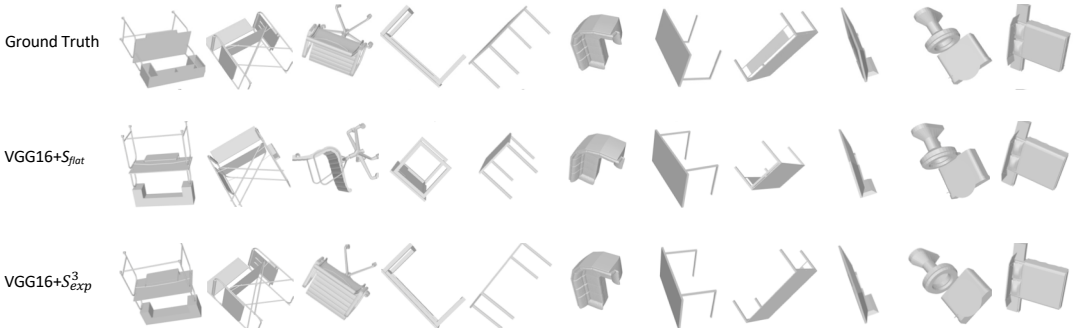


FIGURE A.4: **Visualization of 3D rotation estimation on *ModelNet10-SO3***.

A.4 Derivation of Jacobian for S_{flat} and S_{exp}

First, we provide detailed derivation of Eq. 7 in the main chapter. Given the ℓ_2 normalization form:

$$p_j = g(o_j; \boldsymbol{O}) = \frac{f(o_j)}{\sqrt{\sum_k f(o_k)^2}}$$

with arbitrary univariate mapping $f(\cdot)$, we have:

$$\frac{\partial p_j}{\partial o_i} = \frac{\frac{df(o_j)}{do_i} \cdot A - f(o_j) \cdot \frac{\partial A}{\partial o_i}}{A^2} \tag{A.1}$$

$$= \frac{\frac{df(o_j)}{do_i} \cdot A - f(o_j) \cdot p_i \cdot \frac{df(o_i)}{do_i}}{A^2} \tag{A.2}$$

$$= \frac{1}{A}\left[\frac{df(o_j)}{do_i} - p_i \cdot p_j \cdot \frac{df(o_i)}{do_i}\right] \tag{A.3}$$

$$= \begin{cases} \frac{f'(o_i)}{A} \cdot (1 - p_i \cdot p_j), & \text{when j=i} \\ \frac{f'(o_i)}{A} \cdot (0 - p_i \cdot p_j), & \text{when j}\neq\text{ i} \end{cases} \tag{A.4}$$

where $A = \sqrt{\sum_k f(o_k)^2}$.

Thus the Jacobian matrix of $g : \boldsymbol{O} \to \boldsymbol{P}$ is as follows

$$\mathbf{J}_g = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{O}} = \left[\frac{\partial \boldsymbol{P}}{\partial o_0}, \frac{\partial \boldsymbol{P}}{\partial o_1}, \cdots, \frac{\partial \boldsymbol{P}}{\partial o_n}\right] \tag{A.5}$$

$$= \begin{bmatrix} \frac{\partial p_0}{\partial o_0} & \frac{\partial p_0}{\partial o_1} & \cdots & \frac{\partial p_0}{\partial o_n} \\ \frac{\partial p_1}{\partial o_0} & \frac{\partial p_1}{\partial o_1} & \cdots & \frac{\partial p_1}{\partial o_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial o_0} & \frac{\partial p_n}{\partial o_1} & \cdots & \frac{\partial p_n}{\partial o_n} \end{bmatrix} \tag{A.6}$$

$$= \begin{bmatrix} 1 - p_0 p_0 & -p_1 p_0 & \cdots & -p_n p_0 \\ -p_0 p_1 & 1 - p_1 p_1 & \cdots & -p_n p_1 \\ \vdots & \vdots & \ddots & \vdots \\ -p_0 p_n & -p_1 p_n & \cdots & 1 - p_n p_n \end{bmatrix} \begin{bmatrix} \frac{f'(o_0)}{A} & & & \\ & \frac{f'(o_1)}{A} & & \\ & & \ddots & \\ & & & \frac{f'(o_n)}{A} \end{bmatrix} \tag{A.7}$$

$$= \left(\boldsymbol{I} - \begin{bmatrix} p_0 p_0 & p_1 p_0 & \cdots & p_n p_0 \\ p_0 p_1 & p_1 p_1 & \cdots & p_n p_1 \\ \vdots & \vdots & \ddots & \vdots \\ p_0 p_n & p_1 p_n & \cdots & p_n p_n \end{bmatrix} \right) \begin{bmatrix} \frac{f'(o_0)}{A} & & & \\ & \frac{f'(o_1)}{A} & & \\ & & \ddots & \\ & & & \frac{f'(o_n)}{A} \end{bmatrix} \tag{A.8}$$

A.4.1 $\mathcal{S}_{flat}$ case

In this case, we only take flat ℓ_2 normalization on $\boldsymbol{O}$ to obtain $\boldsymbol{P}$, namely $p_j = g(o_j; \boldsymbol{O}) = \frac{o_j}{\sqrt{\sum_k o_k^2}}$. This means $f(o_i) = o_i$ and $f'(o_i) = 1$. Thus Eq. A.8 becomes:

$$\mathbf{J}_{\mathcal{S}_{flat}} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{O}} \tag{A.9}$$

$$= \left(\boldsymbol{I} - \begin{bmatrix} p_0p_0 & p_1p_0 & \cdots & p_np_0 \\ p_0p_1 & p_1p_1 & \cdots & p_np_1 \\ \vdots & \vdots & \ddots & \vdots \\ p_0p_n & p_1p_n & \cdots & p_np_n \end{bmatrix} \right) \begin{bmatrix} \frac{1}{A} & & & \\ & \frac{1}{A} & & \\ & & \ddots & \\ & & & \frac{1}{A} \end{bmatrix} \tag{A.10}$$

$$= [\frac{\partial \boldsymbol{P}}{\partial o_0}, \frac{\partial \boldsymbol{P}}{\partial o_1}, \cdots, \frac{\partial \boldsymbol{P}}{\partial o_n}] \tag{A.11}$$

$$= (\boldsymbol{I} - \boldsymbol{P} \otimes \boldsymbol{P}) \cdot \frac{1}{A} \tag{A.12}$$

where $\otimes$ denotes outer product.

A.4.2 $\mathcal{S}_{exp}$ case

In this case, we take spherical normalization on $\boldsymbol{O}$ to obtain $\boldsymbol{P}$, namely $p_j = g(o_j; \boldsymbol{O}) = \frac{e^{o_j}}{\sqrt{\sum_k (e^{o_k})^2}}$. This means $f(o_i) = e^{o_i}$ and $f'(o_i) = e^{o_i}$. Thus Eq. A.8 becomes:

$$\mathbf{J}_{\mathcal{S}_{exp}} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{O}} \tag{A.13}$$

$$= \left(\boldsymbol{I} - \begin{bmatrix} p_0p_0 & p_1p_0 & \cdots & p_np_0 \\ p_0p_1 & p_1p_1 & \cdots & p_np_1 \\ \vdots & \vdots & \ddots & \vdots \\ p_0p_n & p_1p_n & \cdots & p_np_n \end{bmatrix} \right) \begin{bmatrix} p_0 & & & \\ & p_1 & & \\ & & \ddots & \\ & & & p_n \end{bmatrix} \tag{A.14}$$

$$= (\boldsymbol{I} - \boldsymbol{P} \cdot \boldsymbol{P}^T) \cdot diag(\boldsymbol{P}) \tag{A.15}$$

$$= (\boldsymbol{I} - \boldsymbol{P} \otimes \boldsymbol{P}) \cdot diag(\boldsymbol{P}) \tag{A.16}$$

where $\otimes$ denotes outer product.

$$\mathbf{J}_{\mathcal{S}_{exp}} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{O}} = (\boldsymbol{I} - \boldsymbol{P} \cdot \boldsymbol{P}^T) \cdot diag(\boldsymbol{P}) \tag{A.17}$$

Appendix B

Supplementary Materials for Quasibinary Classifier

B.1 Proof of Eq. (4.6)

Given q_k as the probability for a sample to be the k-th class, we will prove that the sum of all q_k equals the total number of labels (#label) in the sample (Eq. (6) in the main chapter), *i.e.*:

$$\sum_k q_k = \text{\#label} \tag{B.1}$$

As a quick explanation, we first show a simple example. The mathematical proof is followed after.

Example. Let us consider a 3-class problem, *e.g.* $\{A, B, C\}$ and for a specific sample we have #label=2. We can decompose the Bernoulli probabilities $P(A)$, $P(B)$ and $P(C)$ for each class into the summation of joint probabilities from a set of mutually exclusive joint events. The decompositions are:

$$P(A) = P(AB\overline{C}) + P(A\overline{B}C) \tag{B.2}$$
$$P(B) = P(AB\overline{C}) + P(\overline{A}BC) \tag{B.3}$$
$$P(C) = P(A\overline{B}C) + P(\overline{A}BC) \tag{B.4}$$

Note that #label=2 indicates a sample can only have 2 labels, thus probabilities like $P(A\overline{B}\overline{C})$, $P(ABC)$ are 0, since they correspond to #label=1 and #label=3. As a result, we have

$$P(A) + P(B) + P(C) \tag{B.5}$$
$$= 2 \times (P(AB\overline{C}) + P(A\overline{B}C) + P(\overline{A}BC)) \tag{B.6}$$
$$= 2 \times 1 \tag{B.7}$$
$$= \text{\#label} \tag{B.8}$$

Problem definitions. Let us now consider a K-class problem. We define a binary random variable $Y_k \in \Omega = \{0, 1\}$ as the event for a data sample to associate with the k-th label ($Y_k = 1$) or not ($Y_k = 0$). Thus, $P(Y_k)$ follows Bernoulli distribution, and we have

$$P(\Omega) = p(Y_k = 1) + p(Y_k = 0) = 1 \tag{B.9}$$
$$\text{and} \quad P(Y_k) \in [0, 1], \tag{B.10}$$

where $\Omega = \{Y_k = 0, Y_k = 1\}$.

If we assume $Y_1, \cdots, Y_K$ are independent w.r.t. to each other, we naturally have

$$0 \leq \sum_k^K P(Y_k) \leq K, \quad \text{where } \{Y_k\} \text{ are independent.}$$

But now, given the constraint of #label $= n$, $Y_1, \cdots, Y_K$ are no longer independent. Essentially, Eq. (6) in the main chapter need us to prove:

$$\sum_k^K p(Y_k = 1) = n, \qquad \text{given \#label=n.} \tag{B.11}$$

Proof. We denote the event set $\Phi = \Omega^K$ as the joint probability for a sample to have/not have each of the K labels as:

$$J = (Y_1, \cdots, Y_k, \cdots, Y_K) \in \Phi$$

Since each Y_k has a binary choice and they are mutually exclusive, the joint event space Φ is of size $||\Phi|| = 2^K$, and

$$\sum_{J \in \Phi} P(J) = 1 \tag{B.12}$$

Now, we know a sample only has n labels, *i.e.* #label=n, the non-zero probability of joint event J corresponds to the n-combinations of choosing n classes out of K to have $Y_k = 1$. We denote such a subset of the joint events space as $\Phi(\binom{K}{n})$, which has a size of $\binom{K}{n} = \frac{K!}{n!(K-n)!}$.

$$\sum_{J \in \Phi(\binom{K}{n})} P(J) = 1, \quad \text{when \#label=n.} \tag{B.13}$$

Decomposing binary event $p(Y_k = 1)$ as the summation of the joint probability like equations (B.2),(B.3),(B.4), we observe each $J \in \Phi(\binom{K}{n})$ contributes n times in such a decomposition of each binary event $p(Y_k = 1)$ separately.

Thus, according to equation (B.13), we get:

$$\sum_{k}^{K} p(Y_k = 1) = n \times \sum_{J \in \Phi\binom{K}{n}} P(J) = n \tag{B.14}$$

B.2 More results on One-vs.rest image classification

TABLE B.1: **One-vs.-rest image classification**. Comparison of top-1 error rate on CIFAR10, CIFAR100 and Tiny ImageNet, with the total number of classes being 10, 100, and 200. Binary classifiers are good for small amounts of classes. Quasibinary classifiers are competitive with softmax.

	CIFAR10	**CIFAR100**	**Tiny-ImageNet**
		ResNet18	
Binary classifiers	**4.8**	35.4	×
Quasibinary classifiers (*Ours*)	4.9	**21.9**	**42.9**
Softmax classifiers	5.2	22.2	43.3
		DenseNet40	
Binary classifiers	8.6	×	×
Quasibinary classifiers (*Ours*)	6.7	32.6	55.0
Softmax classifiers	**6.4**	**31.2**	**53.0**
		VGG16	
Binary classifiers	**6.1**	25.8	×
Quasibinary classifiers (*Ours*)	8.3	**25.5**	48.9
Softmax classifiers	8.0	25.8	**48.7**

×: Failed to converge.

Appendix C

Supplementary Materials for Vec2Bundle

C.1 Non-decreasing property of negation rule

In Section 4 of the main chapter, we discuss the problem of balancing accuracy with specificity for multi-label classification. For this purpose, we introduce a negation rule for deriving the probabilities of internal nodes on a class hierarchy given the probabilities of the leaf nodes. Since an internal node corresponds to the event that an image is relevant to at least one of the leaf nodes of such internal node, the higher level internal node on the hierarchy should correspond to a higher probability. Therefore, we need the resulting probability derived from the negation rule to satisfy this property. We now prove the negation rule guarantees $P(A) \geq P(B)$ if node A is the ancestor of node B.

Proof. Let us first rewrite node A and B as a set of their leaf nodes, *i.e.* $A = \{l_i | i \in [1, N]\}$ and $B = \{l_j | j \in [1, N]\}$, where N is the total number of leaf nodes. According to the negation rule, we have:

$$P(A) = 1 - \prod_{l_i \in A} (1 - P(l_i)) \tag{C.1}$$

$$P(B) = 1 - \prod_{l_j \in B} (1 - P(l_j)) \tag{C.2}$$

Since A is the ancestor of B, we have $B \subseteq A$, and as a result $A = B \cup A \backslash B$. Thus, $P(A)$ can be rewritten as:

$$P(A) = 1 - \prod_{l_i \in B} (1 - P(l_i)) \cdot \prod_{l_j \in A \backslash B} (1 - P(l_j)) \tag{C.3}$$

Since probability are all within $[0, 1]$, we have $\prod_{l_j \in A \backslash B} (1 - P(l_j)) \leq 1$. Thus

$$\prod_{l_i \in B} (1 - P(l_i)) \cdot \prod_{l_j \in A \backslash B} (1 - P(l_j)) \leq \prod_{l_i \in B} (1 - P(l_i)) \tag{C.4}$$

Combine Eq. C.2,C.3,C.4, we arrive at $P(A) \geq P(B)$.

C.2 Experimental details

C.2.1 Statistics of hierarchies

We show in Table. C.1 the statistics of class hierarchies that we experimented with in Section 5.3, for balancing accuracy *vs*. specificity of single-label image classification. In comparison, Vec2Bundle hierarchy has a similar or higher height than semantic hierarchies as well as the two learned hierarchies [62, 172]. In terms of the number of internal nodes (#Inter), Vec2Bundle hierarchies have the largest number. Since each internal node corresponds to a different bundle of leaf nodes, Vec2Bundle hierarchies have the richest bundle patterns available for balancing accuracy and specificity. Moreover, we also show the mean and standard deviation of the number of direct children the internal nodes have. These two statistics reflect how fine-grained and balanced the branches on the hierarchy are. As we can see, Vec2Bundle has a mean of 2.0 ($Mean(\#child)$) and a standard deviation of 0.0 ($Std(\#child)$) on all 3 datasets. That is to say, all the internal nodes on the Vec2Bundle hierarchy have exactly two children (*i.e*. binary tree). This is attributed to the hierarchical clustering algorithm we adopted, where a pair of the closest clusters are merged at each clustering step. Subsequently, making prediction to an internal node one level higher by the classifier only sacrifices the least amount of specificity (information gain), which is inversely proportional to the number of leaf nodes of such internal node. We conclude the hierarchical structure of the Vec2Bundle hierarchy makes it suitable for balancing accuracy and specificity in single-label image classification.

C.2.2 Visualization of class-prototype embedding

To have a qualitative view of how the learned embedding from the proposed RBF embedding layer (Section 3) captures the class relationships, we visualize class-prototype embedding with T-SNE [105] in 2D. Fig. C.1 and Fig. C.2 show the results.

Class-prototype embedding for single-label classification We show in Fig. C.1 the 2D visualization of the class-prototypes learned on CIFAR100. Quite a few semantic meaningful local structures can be observed such as {'cup', 'bowl', 'plate'}, {'sunflower', 'tulip', 'poppy', 'rose', 'orchid'} and {'oak_tree', 'pine_tree', 'willow_tree', 'palm_tree'}. Moreover, we also observe a few visually correlated local structures such as {'bicycle', 'motorcycle'}, {'worm', 'snake'} and {'plate', 'clock'}. These local structures reflect the confusion patterns a single-label classifier may experience and thus are potentially helpful for accuracy-specificity trade-off.

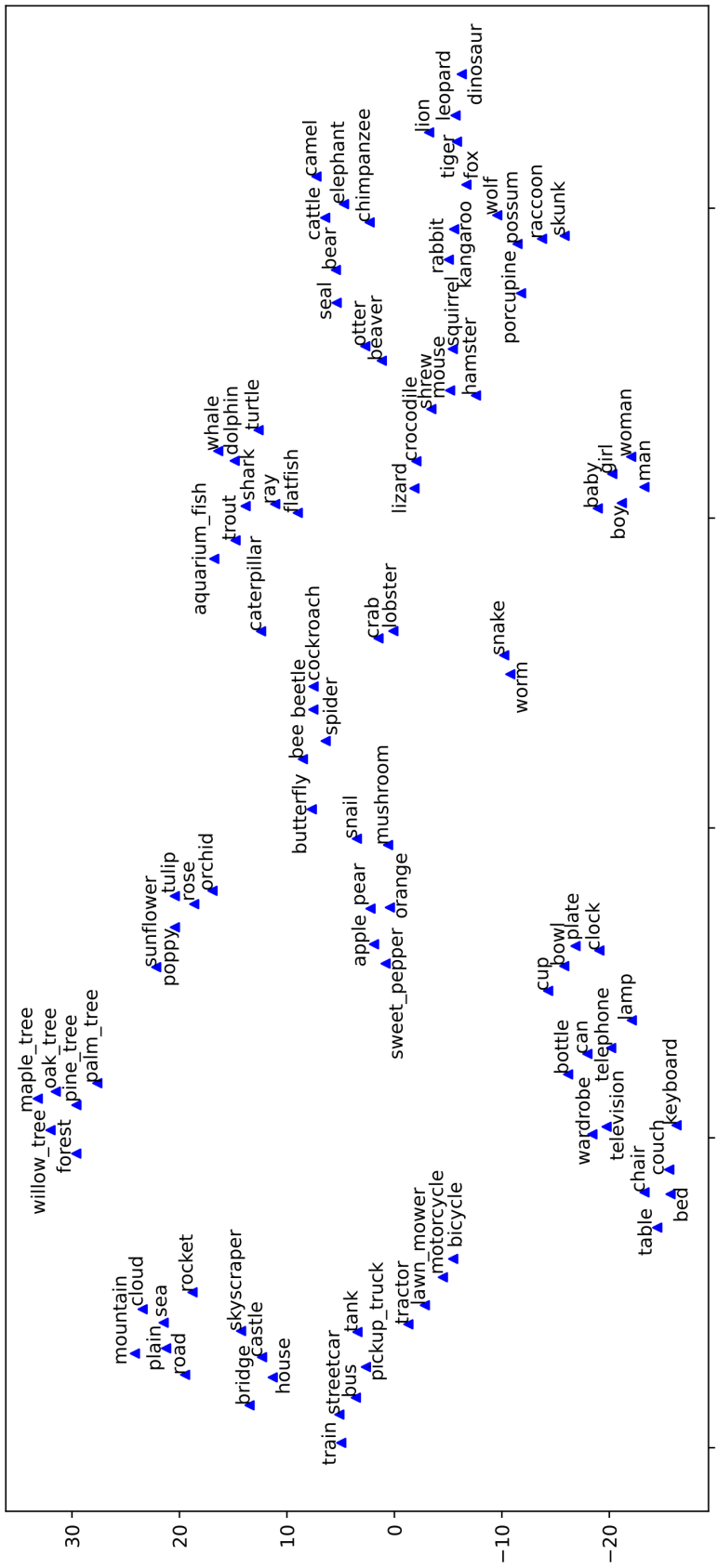


FIGURE C.1: **T-SNE visualization of class-prototypes embedding in 2D on CIFAR100.** Local structures that are semantically meaningful (*e.g.* {'cup', 'bowl', 'plate'}) or visually correlated (*e.g.* {'plate' and 'clock'}) are observed in the embedding space.

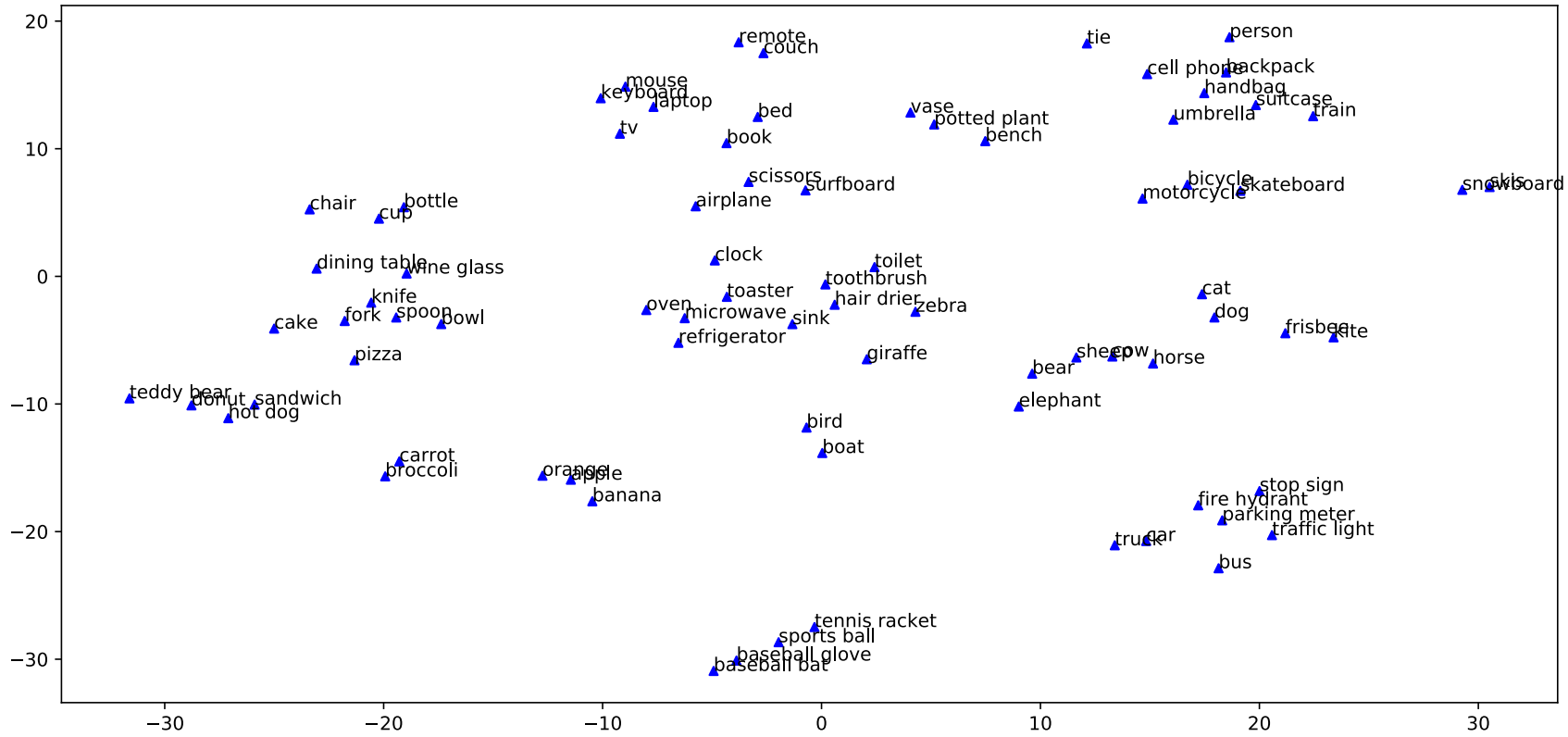


FIGURE C.2: **T-SNE visualization of class-prototypes embedding in 2D on MS-COCO.** Class dependencies in terms of co-occurrence are observed in the embedding space.

CIFAR100	Height	#Inter	Mean(#child)	Std(#child)
Semantic	9	63	2.6	1.4
Hinton [62]	3	11	10.0	5.5
HDcnn [172]	3	11	10.0	25.7
Vec2Bundle (*ours*)	11	99	2.0	0.0
ILSVRC65	Height	#Inter	Mean(#child)	Std(#child)
Semantic	4	8	8.0	9.0
Hinton [62]	3	6	10.3	9.9
HDcnn [172]	3	6	10.3	16.9
Vec2Bundle (*ours*)	11	56	2.0	0.0
ImageNet	Height	#Inter	Mean(#child)	Std(#child)
Semantic	16	372	3.7	3.0
Hinton [62]	3	101	10.9	11.7
HDcnn [172]	3	101	10.9	89.0
Vec2Bundle (*ours*)	16	999	2.0	0.0

TABLE C.1: **Statistics of hierarchies** in Section 5.3. The internal nodes of Vec2Bundle hierarchies have the largest population (#Inter), the smallest number of the mean (Mean(#child)) and standard deviation (Std(#child)) of children.

Class-prototype embedding for multi-label classification We show in Fig. C.2 the 2D visualization of the class-prototypes learned on MS-COCO14. In comparison to the class-prototype learned on single-label classification problem, visually-correlated local structures are less common in this case. Instead, we observe local structures capture co-occurrence relationships are more often, *e.g.* {'cup' and 'bottle'}, {'knife','fork','spoon', 'bowl'}, {'remote', 'couch'}, {'toilet', 'toothbruch'}. This is because an image with multiple class-labels is learned to embed its feature representation $\vec{\mathbf{x}}$ to stay close to multiple class-prototypes at the same time. This in return forces classes that frequently co-occur in the same image to share a local embedding structure.

We conclude the learned class-prototype embedding is able to capture class relationships such as visual-correlation or co-occurrence.

C.2.3 Semantics of the Vec2Bundle hierarchy

Last, we show in Fig. C.3 the 3 class hierarchies we analysed in Section 5.4 on CIFAR100. Namely, a simple semantic hierarchy $H^{(\text{simple})}_{\text{semantic}}$ with 3 levels [82], a more complex semantic hierarchy H_{semantic} [5] and a Vec2Bundle hierarchy $H_{\text{vec2bundle}}$. We observe Vec2Bundle hierarchy $H_{\text{vec2bundle}}$ is as complex as semantic hierarchy H_{semantic}, which allows more pattern of bundles for

balancing accuracy *vs*. specificity. In comparison of the two semantic hierarchies with the Vec2Bundle hierarchy, the internal nodes on $H_{\text{vec2bundle}}$ do not have a semantic names. However, we do find similar hierarchy structures shared between H_{semantic} and $H_{\text{vec2bundle}}$, for example, {'man', 'woman', 'boy','girl', 'baby'}, {'dolphin', 'whale'} and {'orchid', 'poppy', 'sunflower', 'rose', 'tulip'}. This once again justifies the certain amount of semantics carried by Vec2Bundle, as it is suggested by Table 1. in main chapter. Moreover, we also find Vec2Bundle discovers a more fine-grained hierarchical structure than semantic hierarchy. For example, {'bus', 'streetcar', 'train'} are bundled as 'public transport' in the semantic hierarchy, whereas Vec2Bundle hierarchy bundles {'bus', 'streetcar'} first and then merges with 'train'. This enables Vec2Bundle to resolve confusions (*i.e*. between 'bus' and 'streetcar') more effectively without increasing the size of the bundle too much, which explains why Vec2Bundle outperforms semantic hierarchy in balancing accuracy *vs*. specificity trade-off. Last, we also find Vec2Bundle discovers weak semantic structures such as {'bicycle', 'motorcycle'}, {'snake', 'worm'} and {'clock', 'plate'}, which are not on a semantic hierarchy. Although they do not directly correspond to any structure on a semantic hierarchy, however, they capture the frequent visual confusions patterns. In theory, we can attach a semantic name to these bundles such as 'ridable vehicle', 'tube-shaped creature', and 'round-shaped item', we look upon this as the further works. We conclude our Vec2Bundle hierarchies are able to learn a certain amount of semantics.

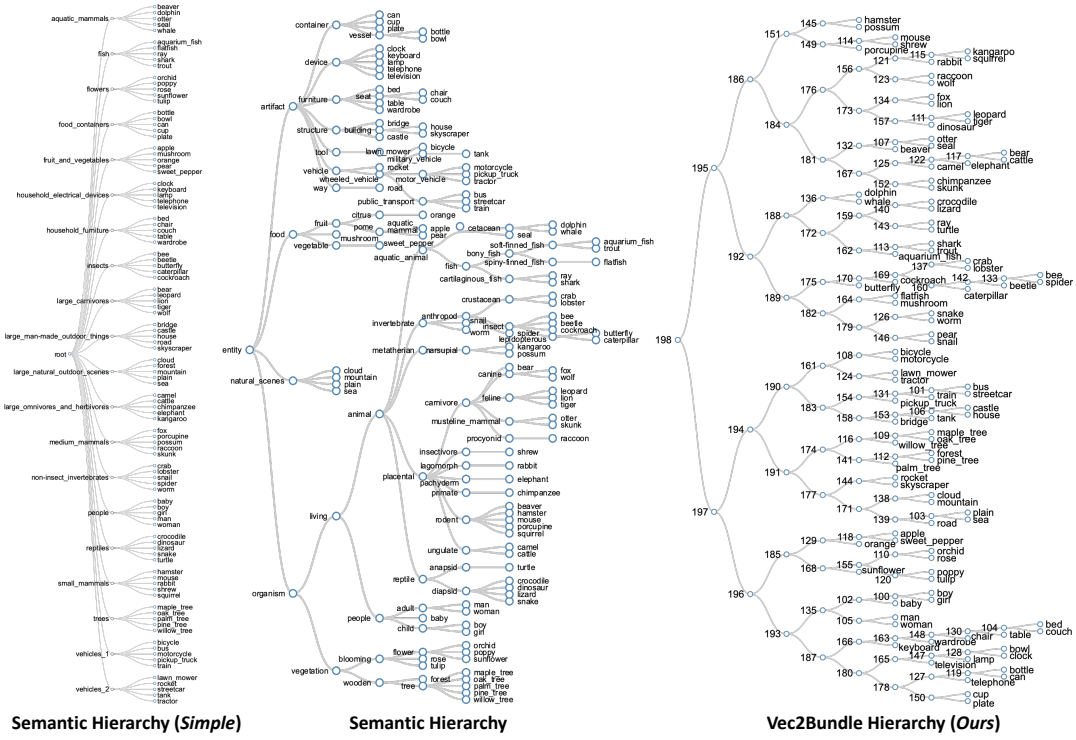


FIGURE C.3: **Comparison of semantic hierarchies *vs.* Vec2Bundle hierarchy** on CIFAR100 shows that the Vec2Bundle hierarchy contains a certain amount of semantics.

Bibliography

[1] Görkem Algan and Ilkay Ulusoy. "Image classification with deep learning in the presence of noisy labels: A survey". In: *Knowledge-Based Systems* (2021).

[2] Jurgen Assfalg, Alberto Del Bimbo, and Pietro Pala. "Retrieval of 3D Objects by Visual Similarity". In: *MIR*. 2004.

[3] Mathieu Aubry et al. "Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models". In: *CVPR*. 2014.

[4] Aayush Bansal, Bryan Russell, and Abhinav Gupta. "Marr revisited: 2d-3d alignment via surface normal prediction". In: *CVPR*. 2016.

[5] Björn Barz and Joachim Denzler. "Hierarchy-based image embeddings for semantic image retrieval". In: *WACV*. 2019.

[6] Samy Bengio, Jason Weston, and David Grangier. "Label embedding trees for large multi-class tasks". In: *NeurIPS*. 2010.

[7] Luca Bertinetto et al. "Making Better Mistakes: Leveraging Class Hierarchies with Deep Networks". In: *CVPR*. 2020.

[8] Lucas Beyer, Alexander Hermans, and Bastian Leibe. "Biternion nets: Continuous head pose regression from discrete training labels". In: *GCVPR*. 2015.

[9] Silvia Biasotti, Simone Marini, Michela Mortara, et al. "An overview on properties and efficacy of topological skeletons in shape modelling". In: *Shape Modeling and Applications, International Conference on*. 2003.

[10] Silvia Biasotti et al. "3D shape matching through topological structures". In: *International conference on discrete geometry for computer imagery*. 2003.

[11] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[12] R. W. Brockett. "Robotic manipulators and the product of exponentials formula". In: *Mathematical Theory of Networks and Systems*. 1984.

[13] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. "COCO-Stuff: Thing and stuff classes in context". In: *CVPR*. 2018.

[14] John Canny. "A computational approach to edge detection". In: *TPAMI* (1986).

[15] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *CVPR*. 2017.

[16] Jonathan L Carrivick, Mark W Smith, and Duncan J Quincey. *Structure from Motion in the Geosciences*. John Wiley & Sons, 2016.

[17] Angel X. Chang et al. "ShapeNet: An Information-Rich 3D Model Repository." In: *CoRR* (2015).

[18] Haomin Chen et al. "Deep Hierarchical Multi-label Classification of Chest X-ray Images". In: *MIDL*. 2018.

[19] Zhao-Min Chen et al. "Multi-label image recognition with graph convolutional networks". In: *CVPR*. 2019.

[20] Christopher Bongsoo Choy et al. "Enriching Object Detection with 2D-3D Registration and Continuous Viewpoint Estimation". In: *CVPR*. 2015.

[21] Tat-Seng Chua et al. "NUS-WIDE: a real-world web image database from National University of Singapore". In: *CIVR*. 2009.

[22] Gabriella Csurka et al. "Visual categorization with bags of keypoints". In: *ECCV workshop*. 2004.

[23] Bin Dai, Shilin Ding, Grace Wahba, et al. "Multivariate bernoulli distribution". In: *Bernoulli* (2013).

[24] Erik B Dam, Martin Koch, and Martin Lillholm. *Quaternions, interpolation and animation*. Datalogisk Institut, Københavns Universitet, 1998.

[25] Jia Deng et al. "Fast and balanced: Efficient label tree learning for large scale object recognition". In: *NeurIPS*. 2011.

[26] Jia Deng et al. "Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition". In: *CVPR*. 2012.

[27] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *CVPR*. 2009.

[28] Jacob Devlin et al. "Fast and robust neural network joint models for statistical machine translation". In: *ACL*. 2014.

[29] Ali Diba et al. "Large Scale Holistic Video Understanding". In: *arXiv preprint arXiv:1904.11451* (2019).

[30] Gilad Divon and Ayellet Tal. "Viewpoint Estimation—Insights & Model". In: *ECCV*. 2018.

[31] Thanh-Toan Do et al. "Real-time monocular object instance 6D pose estimation". In: *BMVC*. 2018.

[32] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *CVPR*. 2015.

[33] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[34] Maura Eduarda and David G Henderson. *Experiencing geometry: On plane and sphere*. Prentice Hall, 1996.

[35] David Eigen and Rob Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture". In: *ICCV*. 2015.

[36] Mark Everingham et al. "The pascal visual object classes challenge: A retrospective". In: *IJCV* (2015).

[37] Luca Falorsi et al. "Explorations in Homeomorphic Variational Auto-Encoding". In: *arXiv preprint arXiv:1807.04689* (2018).

[38] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Convolutional two-stream network fusion for video action recognition". In: *CVPR*. 2016.

[39] Pedro F Felzenszwalb et al. "Object detection with discriminatively trained part-based models". In: *TPAMI* (2009).

[40] Harley Flanders. *Differential Forms with Applications to the Physical Sciences by Harley Flanders*. Elsevier, 1963.

[41] Pierre Foret et al. "Sharpness-Aware Minimization for Efficiently Improving Generalization". In: (2020).

[42] David F Fouhey, Abhinav Gupta, and Martial Hebert. "Data-driven 3D primitives for single image understanding". In: *ICCV*. 2013.

[43] Andrea Frome et al. "Devise: A deep visual-semantic embedding model". In: *NeurIPS*. 2013.

[44] Huan Fu et al. "Deep ordinal regression network for monocular depth estimation". In: *CVPR*. 2018.

[45] Thomas Funkhouser et al. "A Search Engine for 3D Models". In: *ACM Trans. Graph.* (2003).

[46] Jin Gao et al. "Transfer learning based visual tracking with gaussian processes regression". In: *ECCV*. 2014.

[47] Ross Girshick. "Fast r-cnn". In: *ICCV*. 2015.

[48] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *AISTATS*. 2010.

[49] Yunchao Gong et al. "Deep convolutional ranking for multilabel image annotation". In: *arXiv preprint arXiv:1312.4894* (2013).

[50] Alexander Grabner, Peter M Roth, and Vincent Lepetit. "3d pose estimation and 3d model retrieval for objects in the wild". In: *CVPR*. 2018.

[51] Chuan Guo et al. "On calibration of modern neural networks". In: *ICML*. 2017.

[52] Saurabh Gupta et al. "Aligning 3D models to RGB-D images of cluttered scenes". In: *CVPR*. 2015.

[53] David Gurarie. "Symmetries and Laplacians: Introduction to harmonic analysis, group representations and applications". In: *Bull. Amer. Math. Soc* (1993).

[54] Amirhossein Habibian, Thomas Mensink, and Cees GM Snoek. "Videostory: A new multimedia embedding for few-example recognition and translation of events". In: *ACMMM*. 2014.

[55] William Rowan Hamilton. "On quaternions; or on a new system of imaginaries in algebra". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1844).

[56] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[57] Kaiming He et al. "Deep residual learning for image recognition". In: *CVPR*. 2016.

[58] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *ICCV*. 2015.

[59] Kaiming He et al. "Mask r-cnn". In: *ICCV*. 2017.

[60] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem". In: *CVPR*. 2019.

[61] Dan Hendrycks and Kevin Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *ICLR*. 2017.

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

[63] David A Hirshberg et al. "Coregistration: Simultaneous alignment and modeling of articulated 3D shape". In: *ECCV*. 2012.

[64] David Hoag. "Apollo guidance and Navigation: Considerations of apollo imu gimbal lock". In: *Cambridge: MIT Instrumentation Laboratory* (1963).

[65] Seunghoon Hong et al. "Online tracking by learning discriminative saliency map with convolutional neural network". In: *ICML*. 2015.

[66] Gao Huang et al. “Densely connected convolutional networks”. In: *CVPR*. 2017.

[67] Ahsan Iqbal and Juergen Gall. “Level Selector Network for Optimizing Accuracy-Specificity Trade-offs”. In: *ICCV Workshops*. 2019.

[68] Fabian Junkert et al. “Cross-modal Image-Graphics Retrieval by Neural Transfer Learning”. In: *ICMR*. 2017, pp. 330–337.

[69] Frédéric Jurie and Michel Dhome. “Real time 3d template matching”. In: *CVPR*. IEEE. 2001.

[70] HM Kabir et al. “Spinalnet: Deep neural network with gradual input”. In: *arXiv preprint arXiv:2007.03347* (2020).

[71] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. “Tracking-learning-detection”. In: *PAMI* (2010).

[72] Abhishek Kar et al. “Amodal completion and size constancy in natural scenes”. In: *ICCV*. 2015.

[73] Tero Karras et al. “Analyzing and improving the image quality of stylegan”. In: *CVPR*. 2020.

[74] Toshikazu Kato et al. “A sketch retrieval method for full color image database-query by visual example”. In: *ICPR*. 1992.

[75] Will Kay et al. “The kinetics human action video dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

[76] Alex Kendall and Roberto Cipolla. “Geometric loss functions for camera pose regression with deep learning”. In: *CVPR*. 2017.

[77] Valentin Khrulkov et al. “Hyperbolic image embeddings”. In: *CVPR*. 2020.

[78] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *ICLR*. 2014.

[79] Henry E Klugh. *Statistics: The essentials for research*. Psychology Press, 2013.

[80] Yoshinori Konishi et al. “Real-Time 2D/3D Object Detection and Pose Estimation based on Template Matching”. In: ().

[81] Svetlana Kordumova, Thomas Mensink, and Cees G. M. Snoek. “Pooling Objects for Recognizing Scenes without Examples”. In: *ICMR*. 2016.

[82] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[83] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *NeurIPS*. 2012.

[84] Harold W Kuhn and Albert W Tucker. "Nonlinear programming". In: *Traces and emergence of nonlinear programming*. 2014.

[85] Lubor Ladicky, Jianbo Shi, and Marc Pollefeys. "Pulling Things out of Perspective". In: *CVPR*. 2014.

[86] Lubor Ladicky, Bernhard Zeisl, and Marc Pollefeys. "Discriminatively Trained Dense Surface Normal Estimation". In: *ECCV*. 2014.

[87] T Jaya Lakshmi and Ch Siva Rama Prasad. "A study on classifying imbalanced datasets". In: *ICNSC*. 2014.

[88] Kimin Lee et al. "Training confidence-calibrated classifiers for detecting out-of-distribution samples". In: *ICLR* (2018).

[89] Michael S Lew et al. "Content-based multimedia information retrieval: State of the art and challenges". In: *ACM TOMCCAP* (2006).

[90] Bo Li et al. "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs". In: *CVPR*. 2015.

[91] Yuncheng Li, Yale Song, and Jiebo Luo. "Improving pairwise ranking for multi-label image classification". In: *CVPR*. 2017.

[92] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *ICLR*. 2018.

[93] Shuai Liao, Efstratios Gavves, and Cees GM Snoek. "Searching and Matching Texture-free 3D Shapes in Images". In: *ICMR*. 2018.

[94] Shuai Liao, Efstratios Gavves, and Cees GM Snoek. "Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres". In: *CVPR*. 2019.

[95] Shuai Liao et al. "Quasibinary Classifier for Images with Zero and Multiple Labels". In: *ICPR*. 2020.

[96] Joseph J Lim, Aditya Khosla, and Antonio Torralba. "FPM: Fine pose parts-based model with 3d cad models". In: *ECCV*. 2014.

[97] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *ECCV*. 2014.

[98] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear cnn models for fine-grained visual recognition". In: *ICCV*. 2015.

[99] Pengze Liu et al. "Localization guided learning for pedestrian attribute recognition". In: *arXiv preprint arXiv:1808.09102* (2018).

[100] Shaoteng Liu et al. "Hyperbolic Visual Embedding Learning for Zero-Shot Recognition". In: *CVPR*. 2020.

[101] Wei Liu et al. "Ssd: Single shot multibox detector". In: *ECCV*. 2016.

[102] Ziwei Liu et al. "Large-scale long-tailed recognition in an open world". In: *CVPR*. 2019.

[103] Teng Long et al. "Searching for Actions on the Hyperbole". In: *CVPR*. 2020.

[104] David G Lowe. "Object recognition from local scale-invariant features". In: *ICCV*. 1999.

[105] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *JMLR* (2008).

[106] Dhruv Mahajan et al. "Exploring the Limits of Weakly Supervised Pretraining". In: *ECCV*. 2018.

[107] Siddharth Mahendran, Haider Ali, and René Vidal. "3D pose regression using convolutional neural networks". In: *ICCV*. 2017.

[108] Siddharth Mahendran, Haider Ali, and Rene Vidal. "A mixed classification-regression framework for 3D pose estimation from 2D images". In: *BMVC*. 2018.

[109] Oded Maimon and Lior Rokach. "Data mining and knowledge discovery handbook". In: (2005).

[110] Francisco Massa, Renaud Marlet, and Mathieu Aubry. "Crafting a multi-task CNN for viewpoint estimation". In: *BMVC*. 2016.

[111] J Michael McCarthy. *Introduction to theoretical kinematics*. MIT press, 1990.

[112] George A Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* (1995).

[113] Frederic Morin and Yoshua Bengio. "Hierarchical probabilistic neural network language model." In: *Aistats*. 2005.

[114] Arsalan Mousavian et al. "3d bounding box estimation using deep learning and geometry". In: *CVPR*. 2017.

[115] Jinseok Nam et al. "Large-scale multi-label text classification — revisiting neural networks". In: *ECML PKDD*. 2014.

[116] Krystyna Napierala and Jerzy Stefanowski. "Types of minority class examples and their influence on learning classifiers from imbalanced data". In: *JIIS* (2016).

[117] Vladimir Nedović et al. "Stages as models of scene geometry". In: *TPAMI* (2009).

[118] Maximillian Nickel and Douwe Kiela. "Poincaré embeddings for learning hierarchical representations". In: *NeurIPS*. 2017.

[119] Roberto Opromolla et al. "A model-based 3D template matching technique for pose acquisition of an uncooperative space object". In: *Sensors* (2015).

[120] Margarita Osadchy, Yann Le Cun, and Matthew L Miller. "Synergistic face detection and pose estimation with energy-based models". In: *JMLR* (2007).

[121] Wanli Ouyang et al. "Factors in finetuning deep model for object detection with long-tail distribution". In: *CVPR*. 2016.

[122] Onur Ozyesil et al. "A survey of structure from motion". In: *arXiv preprint arXiv:1701.08493* (2017).

[123] Giorgio Patrini et al. "Making deep neural networks robust to label noise: A loss correction approach". In: *CVPR*. 2017.

[124] Hugo Penedones et al. *Improving Object Classification using Pose Information*. Tech. rep. Idiap, 2012.

[125] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. "Deep Directional Statistics: Pose Estimation with Uncertainty Quantification". In: *ECCV*. 2018.

[126] Xiaojuan Qi et al. "Geonet: Geometric neural network for joint depth and surface normal estimation". In: *CVPR*. 2018.

[127] Jesse Read. "Scalable multi-label classification". PhD thesis. 2010.

[128] Jesse Read et al. "Classifier chains for multi-label classification". In: *Machine learning* (2011).

[129] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *CVPR*. 2017.

[130] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *CVPR*. 2016.

[131] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *NeurIPS*. 2015.

[132] Shaoqing Ren et al. "Faster R-CNN: towards real-time object detection with region proposal networks". In: *TPAMI* (2016).

[133] Lawrence G Roberts. "Machine perception of three-dimensional solids". PhD thesis. Massachusetts Institute of Technology, 1963.

[134] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *IJCV* (2015).

[135] Nikolaos Sarafianos, Xiang Xu, and Ioannis A Kakadiaris. "Deep imbalanced attribute classification using visual attention aggregation". In: *ECCV*. 2018.

[136] Johannes L Schonberger and Jan-Michael Frahm. "Structure-from-motion revisited". In: *CVPR*. 2016.

[137] David A Schum. *The evidential foundations of probabilistic reasoning*. Northwestern University Press, 2001.

[138] Chris Seiffert et al. "Resampling or reweighting: A comparison of boosting implementations". In: *ICTAI*. 2008.

[139] Ken Shoemake. "Animating rotation with quaternion curves". In: *SIGGRAPH*. 1985.

[140] Nathan Silberman et al. "Indoor segmentation and support inference from rgbd images". In: *ECCV*. 2012.

[141] Carlos N Silla and Alex A Freitas. "A survey of hierarchical classification across different application domains". In: *Data Mining and Knowledge Discovery* (2011).

[142] Karen Simonyan and Andrew Zisserman. "Two-stream Convolutional Networks for Action Recognition in Videos". In: 2014.

[143] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[144] Irwin Sobel and Gary Feldman. "A 3x3 isotropic gradient operator for image processing". In: *a talk at the Stanford Artificial Project in* (1968).

[145] Hao Su et al. "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views". In: *ICCV*. 2015.

[146] Christian Szegedy et al. "Going deeper with convolutions". In: *CVPR*. 2015.

[147] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *ICML*. 2019.

[148] Johan W. H. Tangelder and Remco C. Veltkamp. "A survey of content based 3D shape retrieval methods". In: *MTAP* (2007).

[149] Ran Tao, Efstratios Gavves, and Arnold W M Smeulders. "Siamese Instance Search for Tracking". In: *CVPR*. 2016.

[150] Du Tran et al. "Learning spatiotemporal features with 3d convolutional networks". In: *ICCV*. 2015.

[151] Shubham Tulsiani and Jitendra Malik. "Viewpoints and keypoints". In: *CVPR*. 2015.

[152] Shimon Ullman. "The interpretation of structure from motion". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* (1979).

[153] Grant Van Horn and Pietro Perona. "The devil is in the tails: Fine-grained classification in the wild". In: *arXiv preprint arXiv:1709.01450* (2017).

[154] Grant Van Horn et al. "The inaturalist species classification and detection dataset". In: *CVPR*. 2018.

[155] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* (2014).

[156] Anran Wang et al. "Multi-modal unsupervised feature learning for RGB-D scene labeling". In: *ECCV*. 2014.

[157] Jiang Wang et al. "Cnn-rnn: A unified framework for multi-label image classification". In: *CVPR*. 2016.

[158] Jingya Wang et al. "Attribute recognition by joint recurrent learning of context and correlation". In: *ICCV*. 2017.

[159] Limin Wang, Yu Qiao, and Xiaoou Tang. "Action recognition with trajectory-pooled deep-convolutional descriptors". In: *CVPR*. 2015.

[160] Xiaolong Wang, David Fouhey, and Abhinav Gupta. "Designing deep networks for surface normal estimation". In: *CVPR*. 2015.

[161] Xiaolong Wang et al. "Non-local neural networks". In: *CVPR*. 2018.

[162] Xiaosong Wang et al. "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases". In: *CVPR*. 2017.

[163] Xiaosong Wang et al. "Tienet: Text-image embedding network for common thorax disease classification and reporting in chest x-rays". In: *CVPR*. 2018.

[164] Joe H Ward Jr. "Hierarchical grouping to optimize an objective function". In: *Journal of the American statistical association* (1963).

[165] Jason Weston, Samy Bengio, and Nicolas Usunier. "Wsabie: Scaling up to large vocabulary image annotation". In: *IJCAI*. 2011.

[166] Wikipedia contributors. *Simplex — Wikipedia, The Free Encyclopedia*. 2021.

[167] Changchang Wu. "Towards linear-time incremental structure from motion". In: *3DV*. 2013.

[168] Fei Wu et al. "Weakly semi-supervised deep learning for multi-label image annotation". In: *IEEE Transactions on Big Data* (2015).

[169] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *CVPR*. 2015.

[170] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond pascal: A benchmark for 3d object detection in the wild". In: *WACV*. 2014.

[171] Tong Xiao et al. "Learning from massive noisy labeled data for image classification". In: *CVPR*. 2015.

[172] Zhicheng Yan et al. "HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition". In: *ICCV*. 2015.

[173] Li Yao et al. "Learning to diagnose from scratch by exploiting dependencies among labels". In: *arXiv preprint arXiv:1710.10501* (2017).

[174] Joe Yue-Hei Ng et al. "Beyond short snippets: Deep networks for video classification". In: *CVPR*. 2015.

[175] Sangdoo Yun et al. "Re-labeling ImageNet: from Single to Multi-Labels, from Global to Localized Labels". In: *arXiv preprint arXiv:2101.05022* (2021).

[176] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[177] Bernhard Zeisl, Marc Pollefeys, et al. "Discriminatively trained dense surface normal estimation". In: *ECCV*. 2014.

[178] Min-Ling Zhang and Zhi-Hua Zhou. "Multilabel neural networks with applications to functional genomics and text categorization". In: *TKDE* (2006).

[179] Yinda Zhang et al. "Physically-based rendering for indoor scene understanding using convolutional neural networks". In: *CVPR*. 2017.

[180] Zhilu Zhang and Mert R Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels". In: *arXiv preprint arXiv:1805.07836* (2018).

[181] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *ICCV*. 2017.

[182] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. "Capturing long-tail distributions of object subcategories". In: *CVPR*. 2014.

[183] Daniel Zwillinger and Stephen Kokoska. *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.

Samenvatting

Deze dissertatie wil een antwoord geven op de vraag "Hoe geometrie beter te gebruiken voor een beter beeldbegrip?". Deze vraag wordt beantwoord vanuit twee invalshoeken, namelijk geometrie in visueel beeldbegrip (Deel I) en geometrie in semantisch beeldbegrip (Deel II). In **Deel I** worden 3D-geometrieën zoals de 3D-vorm, het camerastandpunt, de normaalvector, en de 3D-rotatie bestudeerd. In **Deel II** wordt een nieuw type geometrie in de waarschijnlijkheidsruimte van beeldclassificatie onderzocht, namelijk label-geometrie. Onze bijdragen zijn:

Deel I. Deep learning met 3D-geometrie

Hoofdstuk 2: We stellen een op deep learning gebaseerd model voor dat in staat is om textuurloze 3D-vormen over te brengen naar soortgelijke objecten in 2D natuurlijke beelden. Dit wordt bereikt door het vergelijken van een zoekafbeelding in 2D met opeenvolgende gerenderde 3D-beelden in verschillende posities, tijdens welk proces een deep learning matching-functie wordt gebruikt om de overeenkomsten te beoordelen. Zodra het best overeenkomende gerenderde beeld is gevonden, leiden de parameters die worden gebruikt om het gerenderde beeld te genereren tot een beter begrip van de objectposities in het 2D-beeld.

Hoofdstuk 3: Ons uitgangspunt is de waarneming dat veel continue regressieproblemen in de computer vision worden opgelost door afzonderlijke classificatie. De reden hiervoor is dat classificatie een natuurlijke, d.w.z. waarschijnlijke n-simplex heeft, wat leidt tot stabiele training. Hierdoor geïnspireerd, introduceren wij een sferische regressie die de onbegrensde outputs schetst van diepe neurale netwerken tot n-bollen, door middel van sferische exponentiële mapping, waardoor de regressie van een set 3D-doelen, bv. gezichtspunten, normaalvector en 3D-rotatie, even stabiel wordt als classificatienetwerken.

Deel II. Deep learning met label-geometrie

Hoofdstuk 4: We stellen een nieuw type beeldclassificator voor, quasibinaire classificator genaamd, voor 'one-vs.-rest' classificatie, multi-label classificatie en out-of-distribution classificatie, die voorheen op zichzelf staande oplossingen in één model onderbrengt. Het waarschijnlijkheidsmodel van een quasibinaire classificator wordt gespecificeerd door een label-geometrie die verandert afhankelijk van het aantal labels (d.w.z. nul, één of meer) dat elk monster heeft. De quasibinaire classificator leert een dergelijke labelgeometrie door middel van een gedeelde normalisatiefunctie voor alle klassen en datapunten, en laat zien dat dit niet alleen de complexiteit van het model vereenvoudigt, maar ook de betrouwbaarheid van de betrouwbaarheidsscores verhoogt.

Hoofdstuk 5: We leveren een door visuele data gestuurde methode voor het genereren van klasse-hiërarchie, Vec2Bundle, die wordt geleerd door het integreren van klasse-prototypes. De Vec2Bundle hiërarchie is goedkoop te verkrijgen en blijkt beter te zijn in het afwegen van nauwkeurigheid vs. specificiteit bij beeldclassificatie. Verrassend genoeg blijkt de Vec2Bundle hiërarchie een aanzienlijke hoeveelheid semantiek te bevatten zonder dat het hier instructies voor heeft gekregen. Verder breiden we het huidige model uit met het in evenwicht brengen van nauwkeurigheid en specificiteit tot multi-label beeldclassificatie door een negatieregel voor te stellen die een waarschijnlijkheids-hiërarchie berekent.

Acknowledgments

The journey of a Ph.D. can never be easily finished without the support from so many people. Amongst them, the two most important persons for me are my supervisor Cees G.M. Snoek, and co-supervisor E. (Stratis) Gavves.

First and foremost, I would like to express my gratitude to Cees, for his skillful, patient, and responsible guidance. It is my luck to become his second-generation Ph.D. student, preceded by Dr. Xirong Li, who was his first Ph.D. student, and my Master's supervisor as well. As a supervisor, Cees guided me to be creative, precise, and persuasive in pursuing scientific truth. As a senior researcher, he influenced me to be strategic, focused, modest, and punctual. Particularly, I would like to thank Cees for his generosity in tolerating my occasional obsession and stubbornness.

I owe a deep sense of gratitude to Stratis, who helped me to horn my practical research skills in all aspects. Stratis has always been my idol in academics who manages to work on so many threads, but still gives insightful suggestions. I miss every brainstorming session and every deadline sprint we had. I also appreciate the generosity of Stratis for his patience in listening to my unorganized explanation of each half-cooked idea and waiting for me to catch up with his idea development slowly. Reading through his operations on my papers brings me so much satisfaction, and helps me learn so much about writing.

I thank Arnold Smeulders for his insightful suggestions and thought-provoking questions on my research. I also thank his encouragement and help in chairing SOOS talks.

I thank the committee members for my Ph.D. defense: Prof. C. Sánchez Gutiérrez, Prof. A.W.M. Smeulders, Prof. dr. Th. Gevers, Prof. dr. R.C. Veltkamp, Dr. X. Li, Dr. and A. Ghodrati. Thanks for your time in reading and commenting on my thesis. It is my honor to have you as my esteemed opponent.

I would also like to thank Qualcomm research Netherlands, for supporting my Ph.D. and giving me the internship opportunity. The perception team I interned with treated me so well. Special thanks go to Amir Ghodrati and Amirhossein Habibian, who supervised me during this internship. While it was a pity that the pandemic prevented us from working together in the office, both of them spent plenty of time discussing with me. I appreciate Amir Ghodrati for spending so many off-hours discussions in formulating

the problem, whereas I still miss a few in-person discussions with Amirhossein Habibian in the open air at the Science Park.

I would like to also thank my colleague in Quva-Lab - the daily officemates - Kirill Gavrilyuk, Noureldien Hussein, Berkay Kicanaoglu, Mert Kilickaya, Peter O'Connor, Changyong Oh, Adeel Pervez, Tom Runia, Matthias Reisser, Maurice Weiler, for the interesting discussions and social events. I also thank my colleagues at different locations, Deepak Gupta, Zhengyang Li, Ran Tao.

I thank the ISIS colleagues: Marcel Worring, Pascal Mettes, Andrew Brown, Deepak Gupta, Nanne van Noord, Mehmet Altinkaya, Shuo Chen, Yunlu Chen, Sadaf Gulshad, Tao Hu, Ivan Sosnovik, Zenglin Shi, Fida Thoker, William Thong, David Zhang, Jiaojiao Zhao, Riaan Zoetmulder, Devanshu Arya, Sarah Ibrahimi, Ivan Sosnovik, Gjorgji Strezoski, David Zhang, Mihir Jain, Masoud Mazloom. Having you along the journey brings me so many happy moments. Special thanks also go to Dennis and Virginie, for your support and help every now and then.

I thank my family for the long-lasting support and forgiveness for the absence of the happy time that I should stay accompanied with.