Loss functions for set-valued classification

Camille Garcin

▶ **To cite this version:**

Camille Garcin. Loss functions for set-valued classification. Image Processing [eess.IV]. Université de Montpellier, 2023. English. NNT : 2023UMONS052 . tel-04534631

HAL Id: tel-04534631
https://theses.hal.science/tel-04534631

Submitted on 5 Apr 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En Biostatistique

École doctorale : Information, Structures, Systèmes

Unité de recherche : Institut Montpelliérain Alexander Grothendieck (IMAG), France

Fonctions de perte pour la classification à valeurs d'ensembles

Présentée par Camille Garcin
Le 29 septembre 2023

Sous la direction de Alexis JOLY
et Joseph SALMON

Devant le jury composé de

Mathilde MOUGEOT, Professeur des universités, ENSIIE	Rapporteur
Stéphane CHRÉTIEN, Professeur des universités, Université Lumière Lyon 2	Rapporteur
Sandra BRINGAY, Professeur des universités, Université Paul-Valéry Montpellier 3	Président
Guillaume CHARPIAT, Chargé de Recherche, Université Paris-Saclay	Examinateur
Mohamed HEBIRI, Maître de Conférences, Université Gustave Eiffel	Examinateur
Alexis JOLY, Directeur de recherche, Inria	Directeur
Joseph SALMON, Professeur des universités, Université de Montpellier	Directeur
Maximilien SERVAJEAN, Maître de Conférences, Université Paul-Valéry Montpellier 3	Co-encadrant

UNIVERSITÉ DE MONTPELLIER

Résumé

La dernière décennie a été marquée par l'émergence et l'essor des techniques d'apprentissage profond, conduisant à d'énormes progrès dans la vision artificielle, le traitement du langage naturel et la reconnaissance vocale. Ces avancées sont dues aux améliorations matérielles et aux nouvelles architectures, notamment les réseaux neuronaux convolutifs et les transformers, utilisées dans des applications populaires telles que Siri, Deepl, et ChatGPT. Un défi majeur du domaine de la vision artificielle a été ImageNet, une base de données d'un million d'images réparties en 1 000 classes, utilisée comme référence pour mesurer les performances des modèles. Les premiers résultats affichaient une erreur top-1 de 37,5%, tandis que les meilleurs modèles atteignent maintenant 9% d'erreur top-1. Cependant, ImageNet diffère des situations réelles car elle présente des classes artificiellement équilibrées et peu de similarités entre elles. Pour relever des défis plus réalistes, il est essentiel de se concentrer sur des tâches de catégorisation visuelle à grain fin, impliquant des classes similaires et des distributions déséquilibrées avec des classes rares. Pour ce faire, dans cette thèse nous prendrons pour cas d'étude Pl@ntNet, une application écologique basée sur l'apprentissage coopératif, qui permet aux utilisateurs d'identifier les plantes à partir d'images. Dans un tel contexte d'ambiguïté, les classificateurs multi-classes traditionnels qui ne renvoient qu'une seule proposition de classe ne suffisent pas. C'est pourquoi dans ce manuscrit nous étudierons les classificateurs à valeurs d'ensembles, qui retournent pour chaque image un ensemble de classes possibles. Les classificateurs à valeurs d'ensembles sont utiles s'ils renvoient un nombre restreint de classes pour chaque image. Ainsi, il existe plusieurs contraintes sur la taille des ensembles retournés. Dans cette thèse, nous étudions deux types de contraintes: une contrainte de taille ponctuelle, où le classificateur renvoie exactement K classes candidates pour chaque exemple (classification top-K), et une contrainte sur la taille moyenne des ensembles retournés (classification average-K). Afin d'optimiser ces classificateurs à valeurs d'ensembles, nous introduisons de nouvelles fonctions de perte pour améliorer les performances des modèles d'apprentissage profond, une pour chaque type de contrainte. La fonction de perte pour la classification top-K se base sur une fonction de perte charnière combinée à un lissage de la fonction top-K. Pour la classification average-K, nous proposons un modèle à deux têtes, où une tête est chargée d'identifier des classes candidates pour un example donné, et l'autre tête optimise ces suggestions avec une entropie croisée binaire. Les expériences sont menées sur un jeu de données crée à partir des données de Pl@ntNet, Pl@ntNet-300K, constitué de 306 146 images de plantes avec une forte déséquilibre de classes et des ambiguïtés visuelles importantes. Les résultats montrent que les nouvelles fonctions de perte améliorent significativement les performances par rapport à l'entropie croisée, en particulier dans les situations où l'incertitude est élevée.

Mots-clés – apprentissage profond, apprentissage statistique, classification d'images, classificateurs à valeurs d'ensembles, fonctions de perte

Abstract

The last decade has seen the emergence and growth of deep learning techniques, leading to huge advances in computer vision, natural language processing and speech recognition. These advances are due to hardware improvements and new architectures, notably convolutional neural networks and transformers, used in popular applications such as Siri, Deepl, and ChatGPT. A major challenge in the field of computer vision was ImageNet, a database of one million images divided into 1,000 classes, used as a benchmark to measure model performance. Initial results showed a top-1 error of 37.5%, while the best models now achieve a top-1 error of 9%. However, ImageNet differs from real-life situations in that it features artificially balanced classes and few similarities between them. To address more realistic challenges, it is essential to focus on fine-grained visual categorization tasks, involving similar classes and unbalanced distributions. To this end, in this thesis we will take as a case study Pl@ntNet, an ecological application based on cooperative learning, which enables users to identify plants from images. In such a context of ambiguity, traditional multi-class classifiers, which only return a single proposition, are unable to cope. This is why, in this manuscript, we will study set-valued classifiers, which return a set of possible classes for each image. Set-valued classifiers are useful if they return a restricted number of classes for each image. Thus, there are several constraints on the size of the returned sets. In this thesis, we study two types of constraints: a pointwise constraint, where the classifier returns exactly K candidate classes for each example (top-K classification), and a constraint on the average size of the returned sets (average-K classification). To optimize these set-valued classifiers, we introduce new loss functions to improve the performance of deep learning models, one for each type of constraint. The loss function for top-K classification is based on a hinge loss function combined with a smoothing of the top-K function. For average-K classification, we propose a two-headed model, where one head is responsible for identifying candidate classes for a given example, and the other head optimizes these suggestions with binary cross-entropy. Experiments are carried out on a dataset created from Pl@ntNet data, Pl@ntNet-300K, consisting of 306,146 plant images with high class imbalance and visual ambiguities. The results show that the new loss functions significantly improve performance over cross-entropy, particularly in situations of high uncertainty.

Keywords – deep learning, statistical learning, image classification, set-valued classification, loss functions

Remerciements

Je souhaite d'abord remercier mes trois encadrants de m'avoir donné l'opportunité d'effectuer cette thèse à Montpellier sous leur direction. Merci Joseph pour tes précieux conseils scientifiques. Ta connaissance du domaine et du monde académique a été une véritable chance pour moi. Merci pour les conférences où tu m'as envoyé, pour tes retours pertinents sur mes papiers et tes suggestions abondantes de bibliographie. Ta prévenance envers tes doctorants est très appréciable. Merci Alexis pour ton accueil chaleureux au LIRMM et ta bonne humeur ! Merci d'avoir toujours pris le temps d'échanger avec moi malgré ton emploi du temps bien chargé. Merci pour tes idées foisonnantes et ton enthousiasme débordant. Merci Maximilien pour ta disponibilité, ta réactivité et ton engouement lorsqu'il s'agit de surmonter une difficulté technique, qu'il s'agisse d'une preuve ou d'un bug retors !

Je tiens à remercier les membres du jury pour leur déplacement et l'attention portée à ma présentation et aux questions posées. Je suis particulièrement reconnaissant envers mes deux rapporteurs Stéphane Chrétien et Mathilde Mougeot pour le temps qu'ils ont consacré à évaluer mes travaux.

Merci aux membres de l'équipe Zénith pour tous les moments conviviaux passés ensemble, merci à Mathias et sa bonhomie, à Joaquim, Benjamin, et tous les autres arrivés au fil de l'eau. Merci Titouan d'avoir ouvert la voie pour la classification à valeurs d'ensembles, merci pour ta rigueur, ton intégrité scientifique et pour ta disponibilité à répondre à mes interrogations.

Je me souviens aussi avec plaisir des moments passés à l'IMAG et des nombreux doctorants que j'ai eu la chance d'y rencontrer : Raphaël, Nathan, Tanguy et tous les autres. Je remercie mes co-bureaux Paul et Marien pour l'ambiance sereine et conviviale qui régnait dans notre bureau, pour leur calme et leur amabilité. Merci Marien pour ta vivacité d'esprit et ton intérêt pour réfléchir aux colles que je te posais fréquemment ! Ce fut un véritable plaisir de côtoyer quelqu'un de si brillant ! Je tiens à remercier les assistantes d'équipe pour toute l'aide administrative fournie durant ces trois années, pour leur patience et leur bienveillance : merci Nathalie, Cathy et Sophie.

Merci enfin à mes amis, qui ont su agrémenter ce séjour à Montpellier de leur joie de vivre, de leur gentillesse et de leur humour : merci Nicolas, Elisabeth, Domitille, Anne, Marie, Amelia, Jonathan, Vanessa, Maïlys, Quentin, Camille. Je garderai en mémoire ces beaux moments passés ensemble.

Finalement, merci à ma famille de m'avoir soutenu toutes ces années.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Contents

Notation

L	number of classes
$[L]$	set of all classes $\{1, \cdots, L\}$
$[L]_K$	set of all subsets of $[L]$ of size K
K	Integer in $[L]$
Δ^{L-1}	simplex, $\{\mathbf{p} \in \mathbb{R}^L, \mathbf{p} \geq 0, \sum_{i=1}^L p_i = 1\}$
Δ_K^{L-1}	top-K simplex, $\{\mathbf{p} \in \mathbb{R}^L, 0 \leq \mathbf{p} \leq 1, \sum_{i=1}^L p_i = K\}$
$2^{[L]}$	set of all subsets of $[L]$
$\mathbf{s}$	Vector of scores (logits) in $\mathbb{R}^L$
$\mathcal{X}$	Image input space
$\mathbb{P}$	Proability measure on $\mathcal{X} \times [L]$
$p_l(x)$	class conditional probability $\mathbb{P}(Y = l \vert X = x)$
$\mathbf{p}(x)$	Vector of class conditional probabilities for $x \in \mathcal{X}$, in $\mathbb{R}^L$
$\tau_K(\mathbf{s})$, $\mathbf{s}_{[K]}$	top-K function, returns the K-th largest value of $\mathbf{s}$
$\mathbf{s} \to \mathcal{T}_K(\mathbf{s})$	returns a set containing the indices of the K largest values of $\mathbf{s}$

The last decade has seen the emergence and boom of deep learning techniques. Staggering progress was made in computer vision [Vou+18], natural language processing [Khu+23] and speech recognition [Nas+19]. This progress is largely linked with progress in hardware to execute rapid matrix products [RMN09]. It is also due to the emergence of novel architectures, such as convolutional neural networks [LeC+89] and transformers [Vas+17], that have now been widely adopted. This upheaval has had repercussions on many technological tools (Siri, Deepl, ChatGPT, etc.) used by billions of people around the world.

Since then, the community has always strived to solve ever more challenging problems. The toy tasks that served for the first developments of the domain are now considered solved [Kri09; LeC+98]. In computer vision, ImageNet [Rus+15] has been a key challenge for researchers and practitioners. It is a database of more than 1M images and 1000 classes that was released in 2012. It is a recognized benchmark that has created a lot of emulation to reach ever-better levels of performance. The results in the seminal paper reported a top-1 error of 37.5%. Currently, the best models achieve only 9% top-1 error. While there may still be some improvements to gain, there is a need to turn to more realistic tasks. Indeed, ImageNet was crafted from web images, with classes that have little in common and are organized in a balanced way. It is different from real-world projects that can count several thousands of classes, many of which differ only via subtle variations. This is known as Fine-Grained Visual Categorization: the fine distinction of similar classes [WWC19]. This arises in many situations: the categorization of different species, motorcycle models, architectural styles, etc. In addition, ImageNet was artificially balanced, meaning that all classes approximately share the same number of training examples. Again, this is an ideal setup. In reality, this balance rarely occurs in real life. The Pareto principle [Arn14] is everywhere: a few classes account for the majority of examples. The problem with classical datasets

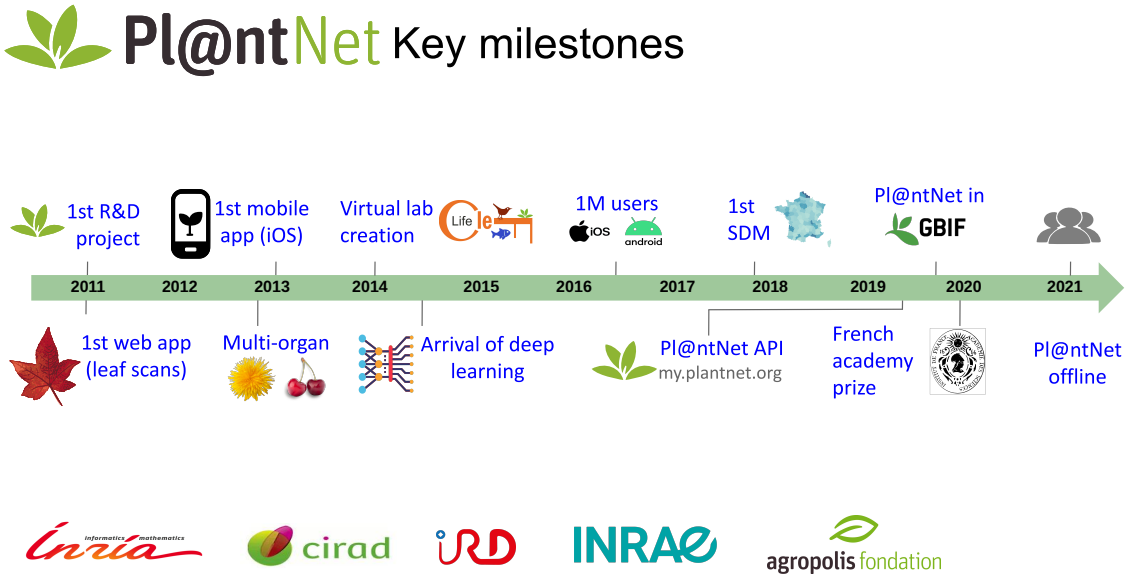


Figure 1.1: *Key milestones of the Pl@ntNet project.*

like ImageNet is that they occult these aspects. When faced with real-world problems, we must deal with similar classes and long-tailed distributions. In this thesis, we will investigate these two aspects:

- How to deal with ambiguous classes?
- Can we obtain good performance in the long-tail, *i.e.*, on classes with few examples?

To tackle this, we need data representative of what happens in real life, with ambiguous and imbalanced classes. Throughout this thesis, we will use data collected by an ecological, crowd-sourced application: Pl@ntNet [Aff+17].

1.1 The Pl@ntNet project

Pl@ntNet is a participatory platform for the production, aggregation and dissemination of botanical observations supported by four founding organizations: CIRAD, INRA, Inria and IRD. Initiated in 2009, the visibility and use of this platform have accelerated since February 2013, with its deployment on mobile platforms (IOS in 2013, and Android in 2014). The application enables users to recognize plants, trees and flowers they encounter in the wild. Since 2013, the Pl@ntNet project has grown impressively: the number of users has doubled every year, reaching more than 20 million in 2022. An impact study concluded that the platform is having a strong societal and economic impact, with 12% of users exploiting it in a professional context in various fields, including agronomy, education, research, commerce and tourism. Figure 1.1 summarizes key milestones for the Pl@ntNet project.

Nowadays, the project counts several engineers, researchers and students to work full time on improving the application. In 2023, the application covers around 43 000 species all over the world. Since the beginning of the project, 800 million plants have

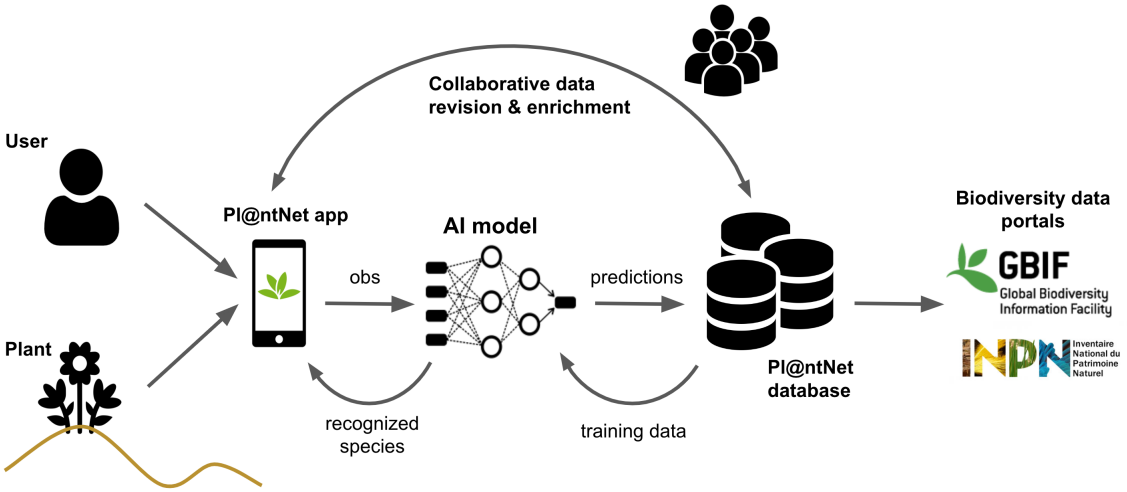


Figure 1.2: *Cooperative learning workflow of Pl@ntNet*

been identified.

The workflow of the application is shown in Figure 1.2. Pl@ntNet is based on a cooperative learning principle. The schematic operation is as follows: first, a Pl@ntNet user generates a plant observation in the field. The observation is then automatically identified by a neural network and some candidate species are returned to the observer for confirmation. All observations are stored in a database and can be reviewed by the community. A user can typically confirm the initial species name shared by the observer (often with the help of the neural network) or propose an alternative species name. Only the observations that reach a sufficient degree of confidence are labeled as valid observations and are added to the training set of the AI model.

The benefits of such a project are numerous. There is of course a pedagogical interest: users get to learn about their environment. Furthermore, Pl@ntNet contributes to the conservation of biodiversity by involving citizens in the collection of data on plant species. Indeed, by encouraging users to upload images of plants they encounter, Pl@ntNet helps create a large-scale database of plant observations. This data provides valuable insights into the distribution and abundance of various plant species, aiding researchers, conservationists, and policymakers in making informed decisions about biodiversity conservation. The data collected by Pl@ntNet is now one of the main sources of plant occurrence data worldwide [Pit+21]. They are integrated in the GBIF infrastructure, the world's largest network providing open access to biodiversity data (funded by governments).

However, with the growing number of species covered by the application, it is getting harder to come up with the true species corresponding to the image, as the number of possible ambiguities between species grows with the total number of classes.

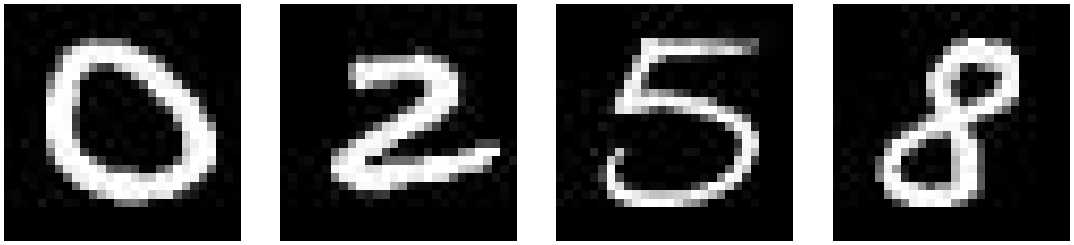


Figure 1.3: *Images from the MNIST dataset: a case with low ambiguity.*

1.2 Set-valued classification

Indeed, consider a classifier $h : \mathcal{X} \to [L]$, where $\mathcal{X}$ is the image input space, L the number of classes (species), and $[L] \coloneqq \{1, \dots, L\}$. Assume the pairs of image/label (X, Y) are generated by an unknown probability distribution $\mathbb{P}$. The risk, or error rate, corresponds to the probability of misclassification:

$$\mathcal{R}(h) \coloneqq \mathbb{P}(Y \neq h(X)) \ . \tag{1.1}$$

The optimal classifier (Bayes classifier) $h^* : \mathcal{X} \to [L]$ returns the most likely class for each input image $x \in \mathcal{X}$:

$$h^*(x) = \underset{l \in \{1,\dots,L\}}{\arg\max}\, \mathbb{P}(Y = l | X = x) \ , \tag{1.2}$$

which results in the Bayes risk:

$$\mathcal{R}(h^*) = \mathbb{P}(Y \neq h^*(X)) = 1 - \mathbb{E}_X[\max_{l \in \{1,\dots,L\}} \mathbb{P}(Y = l|X)] \ . \tag{1.3}$$

We can see that the risk is small whenever the quantities $\max_{l \in \{1,\dots,L\}} \mathbb{P}(Y = l|X = x)$ are large, *i.e.*, when given an image x, one class is much more plausible than the other ones. This happens for easy problems, *e.g.*, when the number of classes is small and the classes have little in common. This case typically corresponds to the MNIST dataset, which is a dataset of handwritten digits where each class represents a digit. The classes are easily distinguishable one from another and ambiguity is quasi non-existent. Some images of the database are shown in Figure 1.3.

However, for real applications like Pl@ntNet, this assumption of low ambiguity does not hold: there are tens of thousands of classes, and many species share important visual traits. This is particularly true for species that belong to the same family. They sometimes differ only by slight variations in leaf shape or color. An example of two similar classes is available in Figure 1.4

On top of that, when someone takes a picture of a plant, it is only a partial representation of that object. The user might zoom into a part of the plant but miss a discriminative feature of this species. For instance, some species differ only by the shape of the leaf, with identical flowers. This is illustrated in Figure 1.5. In that case, if the user only photographs the flower, several species will be possible, and returning a single one will likely result in a mistake.

Figure 1.4: *Examples of visually similar images belonging to two different classes.*

In such cases, the risk becomes high, meaning that even the best possible classifier will achieve low accuracy.

This is problematic for various reasons. For example, an endangered species could be wrongly classified as an invasive species, with dramatic consequences (*e.g.*, uprooting of the plant).

One way to alleviate this issue is to consider classifiers that do not return merely a single class, but a set of candidate classes, *i.e.*, **set-valued** classifiers: $\Gamma : \mathcal{X} \to 2^{[L]}$, where $2^{[L]}$ is the power set of $[L]$. The associated minimization problem then reads:

$$\min_{\Gamma} \ \mathcal{R}(\Gamma) \coloneqq \mathbb{P}(Y \notin \Gamma(X)) \tag{1.4}$$

In practice, this is what Pl@ntNet does: it returns a list of species to the user. Each suggestion has an associated score, with higher scores indicating higher probability.

1.2.1 On the set size

The solution to Equation (1.4) is trivial: it is the classifier that returns all classes for each input image x: $\Gamma(x) = [L]$. This results in a classifier with zero risk. However, such a classifier is completely useless. With this trivial example, we can understand easily that a set-valued classifier is informative only if it returns a few classes. For instance, in the Pl@ntNet context, we must always bear in mind the person that is going to use the application in the wild to recognize a plant. Is it useful if we make more than 100 suggestions? Probably not. On top of that, there are constraints linked to the device: a smartphone can not display dozens of species images on the screen. Therefore, for practical as well as user experience reasons, we are constrained in the set size. For an application like Pl@ntNet, it should not exceed let us say ten candidates.

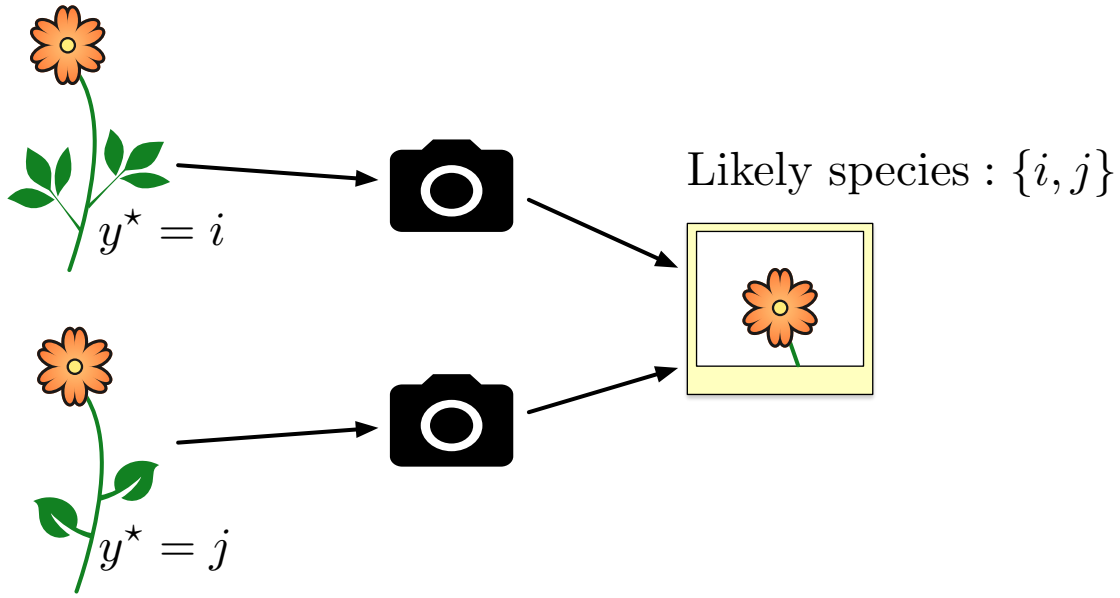


Figure 1.5: *Example of a case with two species differing only by their leaves but having the same flower. If the user takes only a picture of the flower, it is impossible to discriminate between the two species.*

So really the problem we will look into is:

$$\min_{\Gamma} \mathcal{R}(\Gamma) \coloneqq \mathbb{P}(Y \notin \Gamma(X)) \tag{1.5}$$

$$\text{s.t. } \mathfrak{S}(\Gamma) \leq K \ , \tag{1.6}$$

where $\mathfrak{S}(\Gamma)$ denotes a constraint on the size of Γ and K the size constraint. In this thesis, we will consider the case where $K \geq 1$ for the reasons stated above. We will consider K to be an integer, although it can be a real number. In this manuscript, we will study two types of constraint $\mathfrak{S}$: a pointwise size constraint and an average size constraint.

1.2.2 Top-K classification

The first type of constraint we will study is a pointwise constraint:

$$|\Gamma(x)| \leq K, \quad \forall x \in \mathcal{X} \ . \tag{1.7}$$

This is the simplest case: for each input image x, the classifier returns K classes exactly. This is known as top-K classification [LHS15]. We reduce the risk by allowing the classifier to return $K \geq 1$ candidate classes for each example. While this is effective and rather simple to study, it has a drawback: precisely that the classifier returns the same number of classes for each example. Yet, the ambiguity depends on the image. Some images are well zoomed and depict an "easy" species (easily distinguishable from the other ones), in which case a set of size one suffices. Other images are particularly hard to handle and necessitate large sets. It is not a one size fits all.

1.2.3 Average-K classification

To tackle this, we consider another constraint, this time an average size constraint:

$$\mathbb{E}_X|\Gamma(X)| \leq K, \ \ \forall x \in \mathcal{X} \ . \tag{1.8}$$

This is known as average-K classification [DH17]. The constraint is on the average size of the classifier. This allows for flexibility over the previous setting. Indeed, in this setup, the classifier is allowed to return variable-sized sets. This allows to return more classes when needed, and less for non-ambiguous samples.

1.2.4 Relevance of set-valued classifiers?

The risk of (1.5) is lower than that of (1.1) since we now allow the classifier to return multiple classes. So what is the catch? What is the price if we get less error? Well, we lose in informativeness. Although returning a single class will result in a high risk, it has the merit of clarity and simplicity. We provide an answer to the query, the user is provided with a species name. It might not be the correct one, but maybe the user will not notice it. This is not a very satisfactory solution, however. We feel that the set solution is more appropriate. However, if we simply return a set of size *e.g.*, three with three species names, this is not satisfactory either. The user wants to have a clue as to what the true species is. This can be solved by adding extra information on the candidate classes. In Pl@ntNet, for instance, there is a similarity search procedure that displays, for each species suggestion, the images of that species most similar to the query image. This is illustrated in Figure 1.6. This way, the user has additional material to make his decision.

Notice that the traditional multi-class classifier $\Gamma : \mathcal{X} \to [L]$ corresponds to the particular case of a set-valued classifier that always returns sets of size one.

1.2.5 Estimation of set-valued classifiers

In practice, we can not solve (1.5). The reason is that we do not know $\mathbb{P}$. Instead, we have access to n_{train} *i.i.d.* samples $(x_1, x_1), \ldots, (x_{n_{train}}, y_{n_{train}})$ drawn from $\mathbb{P}$. From these samples, we want to train a classifier to predict a relevant set of classes given a new image x. How to proceed? We first need to choose what kind of classifier to consider. In this thesis, we will consider deep learning models for their remarkable performances in computer vision tasks.

For now, let us assume that we have a trained deep model $\hat{\mathbf{f}}$ that predicts a vector of scores $\hat{\mathbf{f}}(x) \in \mathbb{R}^L$ given an image $x \in \mathcal{X}$. How can we go from there to predict a set of candidate classes for input x? There is a natural plug-in way to construct these sets once we have an estimation of the scores (or equivalently of the conditional probabilities). In the top-K case, it suffices to return the top-K largest estimated scores (or equivalently conditional probabilities):

$$\hat{\Gamma}(x) = \mathcal{T}_K(\hat{\mathbf{f}}(x)) \ , \tag{1.9}$$

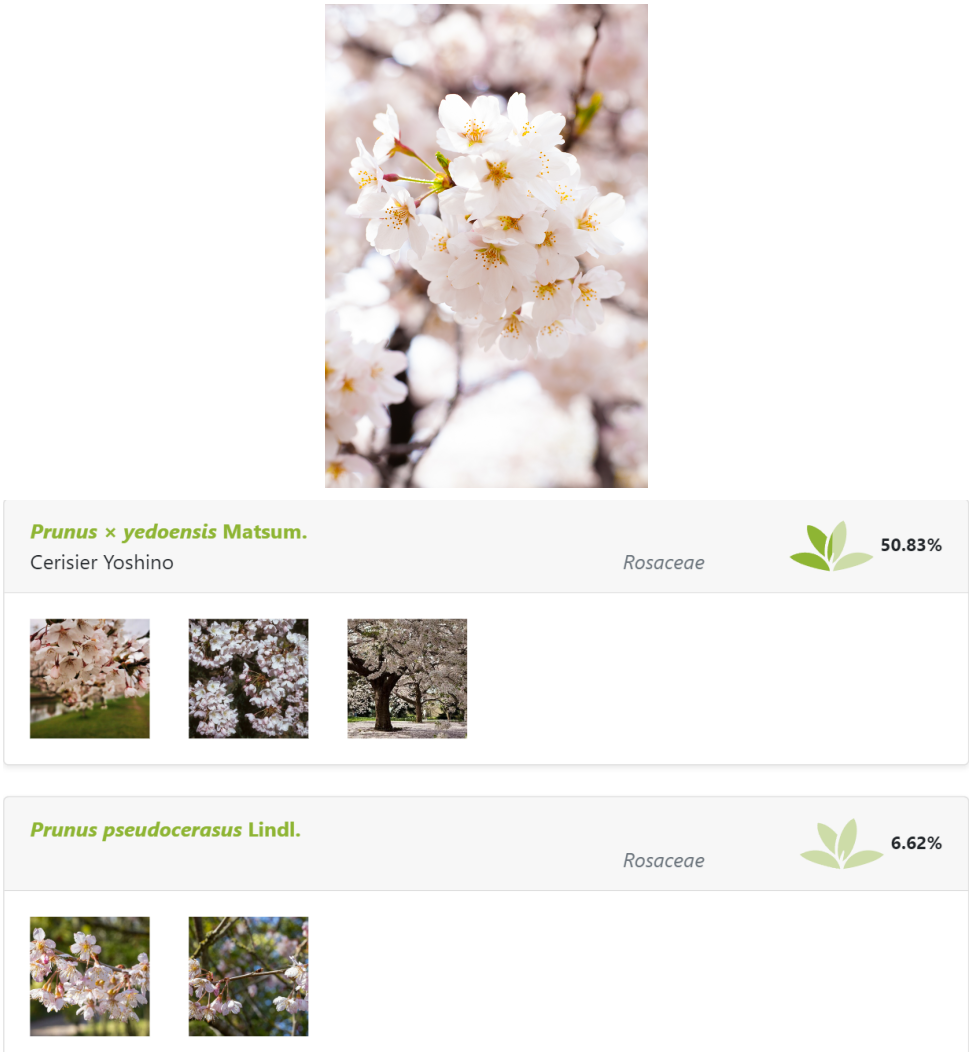


Figure 1.6: *Example of how Pl@ntNet adds information along with the proposed sets. The image above is the query image. Below, the application makes two suggestions of species for the query. For each suggestion, it displays the images of that class most similar to the query image.*

where $\mathcal{T}_K : \mathbb{R}^L \to [L]_K$ is the operator that returns a set containing the K indices corresponding to the K largest value of a vector.

For the average-K case, the set predictor is obtained by thresholding the conditional probabilities on a calibration set, where the threshold is chosen to satisfy the average size constraint. We will detail this later.

The main question resides in how to learn to predict relevant sets, *i.e.*, sets that contain the ground truth label. This translates into the following question: how to train the model? The recipe for training a deep learning model is to use a loss function $\ell : \mathbb{R}^L \times [L] \to \mathbb{R}^+$, compute its gradient w.r.t. the predictions with backpropagation, and update the weights with SGD for instance. The most standard loss function to use is the cross-entropy loss. In this manuscript, the main question we try to answer is the following:

- Is it possible to design loss functions that perform better than cross-entropy on real-world datasets (long-tailed distribution + class ambiguity) for set-valued classification?

1.3 Organization and contributions

The organization of the thesis is as follows:

In Chapter 2, we first list the topics related to the work presented in this manuscript. We then present an overview of set-valued classification methods before explaining the difference between conformal prediction and our work. Then, we derive the expressions of the Bayes top-K and average-K classifiers and explain how to estimate and evaluate such classifiers. We then explain the difference between top-K and average-K classification. Finally, we introduce the notion of surrogate loss functions and top-K and average-K consistency.

In Chapter 3, we present Pl@ntNet-300K, a novel image dataset with high intrinsic ambiguity and a long-tailed distribution built from the database of Pl@ntNet citizen observatory. It consists of 306 146 plant images covering 1 081 species. We highlight two particular features of the dataset, inherent to the way the images are acquired and to the intrinsic diversity of plants morphology: (i) the dataset has a strong *class imbalance*, *i.e.*, a few species accounts for most of the images, and, (ii) many species are *visually similar*, rendering identification difficult even for the expert eye. These two features make the present dataset well-suited for the evaluation of set-valued classification methods and algorithms. Therefore, we recommend two set-valued evaluation metrics associated with the dataset (*macro-average top-K accuracy* and *macro-average average-K accuracy*) and we provide baseline results established by training deep neural networks using the cross-entropy loss. We will present its main characteristics. This dataset will be the main support for our subsequent experiments.

In Chapter 4, we introduce a stochastic top-K hinge loss inspired by recent developments on top-K calibrated losses. Our proposal is based on the smoothing of the top-K operator building on the flexible "perturbed optimizer" framework. We show that our loss function performs very well in the case of balanced datasets while benefiting from a significantly lower computational time than the state-of-the-art top-K loss function. In addition, we propose a simple variant of our loss for the imbalanced case. Experiments on Pl@ntNet-300K show that our loss function significantly outperforms other baseline loss functions.

In Chapter 5, we propose an approach for the much challenging average-K classification. A simple method to solve this task is to threshold the softmax output of a model trained with the cross-entropy loss. This approach is theoretically proven to be asymptotically consistent, but it is not guaranteed to be optimal for a finite set of samples. In this paper, we propose a new loss function based on a multi-label classification head in addition to the classical softmax. This second head is trained using pseudo-labels generated by thresholding the softmax head while guaranteeing that K classes are returned on average. We show that this approach allows the model to better capture ambiguities between classes and, as a result, to return more consistent sets

of possible classes. Experiments on two datasets from the literature demonstrate that our approach outperforms the softmax baseline, as well as several other loss functions more generally designed for weakly supervised multi-label classification. The gains are larger the higher the uncertainty, especially for classes with few samples (this work is currently under review: [Gar+23])

Finally, we conclude in Chapter 6. We take this opportunity to present work in progress on a multi-label loss that has been inspired by work on set-valued classification. This loss function is based on the orthogonal projection onto the top-K simplex. It has good theoretical properties, which we shall present briefly.

1.4 Publications

The work presented in this thesis led to two publications and one paper currently under review:

- Camille Garcin et al. "Pl@ntNet-300K: a plant image dataset with high label ambiguity and a long-tailed distribution". In: *NeurIPS Datasets and Benchmarks 2021*. 2021

- Camille Garcin et al. "Stochastic smoothing of the top-K calibrated hinge loss for deep imbalanced classification". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 7208–7222

- Camille Garcin et al. *A two-head loss function for deep Average-K classification*. Tech. rep. 2023

CHAPTER 2

POSITIONING AND STATE-OF-THE-ART

Contents

2.1 Related work

There are a certain number of topics that are related to set-valued classification that we will not explore in this manuscript. They are presented here for the sake of completeness.

Curriculum learning

Curriculum learning (CL) is a field of machine learning that aims at optimizing the order in which the examples are fed to the model. In general batch gradient descent, the order is random. CL has its roots in human behavior, where learning starts with basic concepts/notions and then progressively extends to harder tasks. The challenge is then to estimate each example's difficulty and find a suitable schedule. The idea originated in [Ben+09]. A comprehensive survey can be found in [WCZ21]. Since Pl@ntNet comprises examples of various difficulties (*e.g.*, well zoomed, easy species and blurry, ambiguous species), it would be an interesting object of study for CL techniques. However, this is an avenue we will not be exploring in this manuscript.

Classification with reject option

Classification with reject option [HW06; CDM16] allows the classifier to abstain from making a decision for a fraction of inputs. In addition to the traditional classifier $h : \mathcal{X} \to [L]$, there is a rejector $r : \mathcal{X} \to \{0, 1\}$. The global prediction consists of the prediction of h if the rejector r does not reject the sample, otherwise, no prediction is made.

While this allows us to reduce the risk by discarding hard examples, this approach in our case is unsatisfactory for several reasons:

- This may lead the classifier to abstain from answering for challenging instances, which are arguably the most interesting ones. Indeed, in the case of highly similar species, the model may hesitate between multiple species, not returning a strong signal for a single one. In the case of a reject option, the model might reject such an example. Instead, in set-valued classification, the classifier will return several species for that example. With the similarity search engine and additional information *e.g.*, the web, the user can then make up his mind to determine which is the true species among the proposed set.
- From a user experience point of view, it is best to return suggestions regardless of their relevance.

In fact, the Pl@ntNet pipeline already includes a rejector. It allows to reject images that are not plants or of poor quality. In Chapter 3, the dataset is constructed with filtered images. Thus, it contains only relevant pictures of good quality, for which we want to make a prediction. Therefore, in the rest of the thesis, we will consider classifiers that return a set of species for all input images x.

Hierarchical classification

Hierarchical classification [Gor87; SF11b] deals with classification when there is a structure on the labels, typically a tree structure. This is true for most problems

where labels are organized in a certain way. For instance, CIFAR100 counts 100 classes that are divided into 20 superclasses (*e.g.,* flowers, people, reptiles) each containing 5 classes. In ImageNet a tree structure also exists between the labels although it is much more complex than CIFAR100. In the case of plant species, there exists a hierarchy established by botanists: species are grouped into genera, which are organized into families…This hierarchy respects biological realities: species belonging to the same genus derive from a common ancestor and usually share important morphological features.

Several works [DKS04; DC00; KS97; CB+04] have proposed methods and algorithms to use this structure on labels. Although Pl@ntNet lends itself well to hierarchical classification, this an avenue we will not pursue in this thesis.

Multi-label classification

Set-valued classification should not be confused with multi-label classification. Both settings are different. The difference resides in the label space $\mathcal{Y}$. For multi-label classification [Dem+12; Liu+21; ZZ14], several labels may be associated with an image x: $\mathcal{Y} = 2^{[L]}$, whereas in set-valued classification a single class is present on the image: $\mathcal{Y} = [L]$. For instance, in the Pl@ntNet application, the users are incentivized to take a clear, zoomed picture of a single plant. This is illustrated in Figure 2.1. Again, we stress that throughout this thesis, only a single label will be associated to each input image x. In particular, all our experiments will be on multi-class datasets, *i.e.,* datasets for which a single label is associated with an image. This is different than multi-label datasets [Lin+14; Eve+10].

2.2 Set-valued classication

2.2.1 Overview

Set-valued classification [Chz+21] relates to classifiers $\Gamma : \mathcal{X} \to 2^{[L]}$ that return a set of classes instead of a single class in the context of multiclass classification. There are two key fundamental properties of a set-valued classifier: its -output- size and its risk. There is a tradeoff to make: increasing the size of the classifier will lower its risk but at the cost of informativeness. There are two "dual" ways to look at the problem. Either we minimize the risk with a constraint on the set size, or we minimize the set size with a constraint on the risk. Let us introduce some notation pertaining to these two properties. For a set-valued classifier Γ and an input $x \in \mathcal{X}$, we define:

$$\mathscr{S}(\Gamma, x) = |\Gamma(x)| \text{ point-wise size} \tag{2.1}$$
$$\mathscr{S}(\Gamma) = \mathbb{E}_X|\Gamma(X)| \text{ average size} \tag{2.2}$$
$$\mathcal{R}(\Gamma, x) = \mathbb{P}(Y \notin \Gamma(X)|X = x) \text{ point-wise error} \tag{2.3}$$
$$\mathcal{R}(\Gamma) = \mathbb{P}(Y \notin \Gamma(X)) \text{ average error} \tag{2.4}$$

Several set-valued formulations exist, depending on the type of constraint and error. We summarize these formulations in Table 2.1. We refer the reader to [Chz+21] for an overview of these methods. The authors give the Bayes classifier for each formulation.

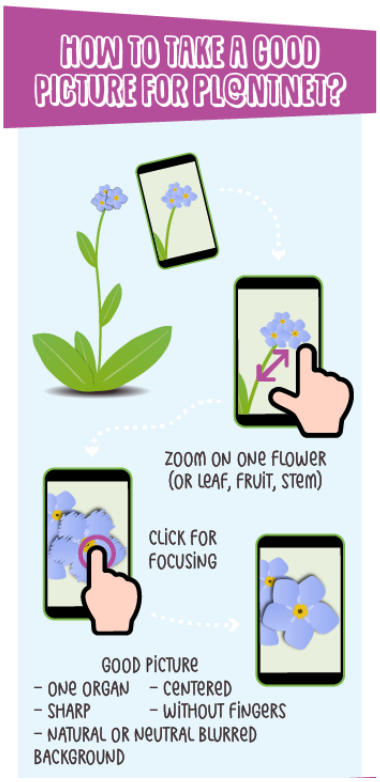


Figure 2.1: *Instruction on how to take a picture in Pl@ntNet. The users are asked to take a zoomed picture of a single specimen.*

In this manuscript, we only focus on the point-wise size and average-size formulations, also known as top-K and average-K classification respectively. The reason is that in the scope of the Pl@ntNet project, as already mentioned, we need control over the size of the predicted sets. The point-wise error and average error do not provide such guarantees and can yield very large sets for some examples. However, some applications require guarantees on the risk (error rate) and would rather have large sets but with a controlled risk. For instance, in disease diagnosis [ALS22; Bat+20], it is best to return a large set of possible diseases than to miss a potentially fatal one. Note that no formulation is better than others. Instead, they correspond to different use cases, and one should choose the most appropriate given the problem at hand.

2.2.2 Conformal prediction

Conformal prediction [VGS05; SV08; FZV23] can also be counted in the set-valued classification methods, as it returns a set of classes for each example. The aim of conformal prediction is to return a set of classes for a test example, such that the probability of the true class to be in the set is larger than a user-defined threshold. Let us describe here the recipe of conformal prediction. Assume we have a model $\hat{\mathbf{p}}$ that gives an estimate of the class conditional probabilities $\hat{\mathbf{p}}(x) \in \mathbb{R}^L$ given an example $x \in \mathcal{X}$. Typically, $\hat{\mathbf{p}}(x)$ is the softmax output of a deep learning model trained with cross-entropy. The idea is to calibrate the output of the model. To this end, we use a set of n input/label pairs, unseen during training, $(X_1, Y_1), \ldots, (X_n, Y_n)$. Then, a

Table 2.1: *Summary of the different set-valued classifications formulations. We recall that* $\mathscr{S}(\Gamma, x) = |\Gamma(x)|$, $\mathscr{S}(\Gamma) = \mathbb{E}_X|\Gamma(X)|$, $\mathcal{R}(\Gamma, x) = \mathbb{P}(Y \notin \Gamma(X)|X = x)$ *and* $\mathcal{R}(\Gamma) = \mathbb{P}(Y \notin \Gamma(X))$.

Name	Minimization objective	Constraint	References
Point-wise size	$\mathcal{R}(\Gamma)$	$\mathscr{S}(\Gamma, x) \leq K$	[LHS15; LHS16; LHS17]
Average size	$\mathcal{R}(\Gamma)$	$\mathscr{S}(\Gamma) \leq K$	[DH17]
Point-wise error	$\mathscr{S}(\Gamma)$	$\mathcal{R}(\Gamma, x) \leq \epsilon$	[FB+21; Lei14; LW14; RSC20]
Average error	$\mathscr{S}(\Gamma)$	$\mathcal{R}(\Gamma) \leq \epsilon$	[SLW19]

calibration score quantifying the uncertainty of the true label is computed for each example. For instance, the score can be computed as $c_i \coloneqq 1 - \hat{\mathbf{p}}(X_i)_{Y_i}$ *i.e.*, one minus the softmax output of the true class. We then define $\hat{q}$ to be the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ empirical quantile of $c_1, \dots c_n$. Finally, given a test example X_{test}, the predicted set for X_{test} is defined to be: $\mathscr{S}(X_{test}) \coloneqq \{j, \hat{p}_j(X_{test}) \geq 1 - \hat{q}\}$. With this simple procedure, we obtain the following result:

$$1 - \alpha \leq \mathbb{P}(Y_{test} \in \mathscr{S}(X_{test})) \leq 1 - \alpha + \frac{1}{n+1} \tag{2.5}$$

Notice that this result holds regardless of whether $\hat{p}$ estimates the conditional probabilities $\mathbb{P}(Y = l|X = x)$ well or not. Equation (2.5) means that we can guarantee the true class lies in the returned set with a level of confidence desired by the user. Conformal prediction allows to return sets of variable size, as the average-K setting. However, much like the point-wise error and average error setting evoked earlier, we have no guarantee on the set size. Again, in the context of Pl@ntNet, this is not a viable solution. However, conformal prediction appears like a pertinent choice for applications that need strict control of the risk, as Equation (2.5) gives a strong certificate. Finally, notice that for the calibration set, we need annotated data. Indeed, we use both the input X_i and its label Y_i. Annotated data can be costly when it requires the careful examination of one or several experts *e.g.*, in medical applications. Instead, in average-K classification, the "calibration step" for the estimation of the threshold does not necessitate labeled data. This is a significant advantage since unannotated data often comes in plentiful and cheap.

We only presented here the application of conformal prediction to classification as this thesis deals with set-valued classification. However, conformal prediction is a broader framework that could also be applied to regression problems for instance.

2.2.3 Top-K classification

Bayes classifier.

Top-K classification [LHS15] corresponds to the case where, for each image, K suggestions are allowed. The problem becomes:

$$\min_{\Gamma} \mathbb{P}(Y \notin \Gamma(X)) \tag{2.6}$$

$$\text{s.t. } |\Gamma(X)| \leq K \tag{2.7}$$

The Bayes classifier corresponds to the classifier that returns the K most probable classes. Indeed, we have:

$$\begin{aligned}
\mathbb{P}(Y \notin \Gamma(X)) &= 1 - \mathbb{P}(Y \in \Gamma(X)) && (2.8)\\
&= 1 - \mathbb{E}(\mathbb{1}(Y \in \Gamma(X))) && (2.9)\\
&= 1 - \mathbb{E}_X \mathbb{E}_{Y|X}(\mathbb{1}(Y \in \Gamma(X))) && (2.10)\\
&= 1 - \mathbb{E}_X \left[\sum_{l=1}^{L} \mathbb{1}(l \in \Gamma(X)) p_l(X) \right] \ , && (2.11)
\end{aligned}$$

which is minimized by predicting the K indices corresponding to the top-K largest conditional probabilities for every $x \in \mathcal{X}$:

$$\Gamma^*(x) = \mathcal{T}_K(\mathbf{p}(x)) \ , \tag{2.12}$$

where $\mathbf{p}(x) \in \mathbb{R}^L$ is the vector of conditional probabilities *i.e.*, $p_l(x) = \mathbb{P}(Y = l | X = x)$

Estimation.

In practice, the conditional probabilities $p_l(x)$ are unknown and one ought to estimate them, typically by training a deep neural network with the cross-entropy loss. With the estimated conditional probabilities $\hat{p}_l(x)$, we can then naturally define the plug-in estimator:

$$\hat{\Gamma}(x) = \mathcal{T}_K(\hat{\mathbf{p}}(x)) \ , \tag{2.13}$$

where $\hat{\mathbf{p}}(x) \in \mathbb{R}^L$ is the vector of estimated conditional probabilities.

Note that (2.13) and (2.12) coincide when the top-K largest estimated conditional probabilities match the top-K largest true conditional probabilities. In particular, the order in which the estimated conditional probabilities are ranked does not matter, as long as they are in the top-K predictions. Note that Chapter 4 will present a method that does not require to estimate the conditionnal probabilities as this could be tricky in the context of high noise and little data. Instead, we work directly with the scores (or logits) and try to place the score of the true label in the top-K scores (*i.e.*, the top-K highest scores).

Evaluation.

Given a test set $(x_1, y_1), \ldots, (x_{n_{test}}, y_{n_{test}})$, we can evaluate a top-$K$ classifier by computing the top-K accuracy. Formally:

$$\textit{Top-K accuracy} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \mathbb{1}\left[y_i \in \mathcal{T}_K(\hat{\mathbf{p}}(x_i)) \right] \tag{2.14}$$

Compared to a multi-class classifier $h : \mathcal{X} \to [L]$ with risk $\mathbb{P}(Y \neq h(X))$, top-K classification yields a top-K classifier $\Gamma : \mathcal{X} \to [L]_K$ with risk $\mathbb{P}(Y \notin \Gamma(X))$ Therefore, top-K classification lowers the risk by predicting K classes for each example. This is already an interesting study case that we will tackle in Chapter 4. However, this approach presents a shortcoming: it treats each example the same way by returning K classes for each example, regardless of the difficulty of each example. The next setting allows us to take that ambiguity into account.

2.2.4 Average-K classification

Bayes classifier.

Average-K classification [DH17] has a constraint looser than top-K classification: the constraint on the set size is not anymore on each example, but rather a global constraint on the average set size:

$$\min_{\Gamma} \mathbb{P}(Y \notin \Gamma(X)) \tag{2.15}$$

$$\text{s.t. } \mathbb{E}_X|\Gamma(X)| \leq K \tag{2.16}$$

This constrained optimization problem has the same solution as the following minimization problem [Ha97]:

$$\min_{\Gamma} \mathbb{P}(Y \notin \Gamma(X)) + \lambda \mathbb{E}_X|\Gamma(X)| \ , \tag{2.17}$$

where the relation between K and λ is derived next. Note that:

$$\mathbb{P}(Y \notin \Gamma(X)) + \lambda \mathbb{E}_X|\Gamma(X)| = \mathbb{E}_{X,Y}(\mathbb{1}[Y \notin \Gamma(X)] + \lambda|\Gamma(X)|) \tag{2.18}$$

$$= \mathbb{E}_X \mathbb{E}_{Y|X}(\mathbb{1}[Y \notin \Gamma(X)] + \lambda|\Gamma(X)|) \tag{2.19}$$

$$= \mathbb{E}_X \sum_{l=1}^{L} p_l(X)\mathbb{1}[Y \notin \Gamma(X)] + \lambda|\Gamma(X)| \tag{2.20}$$

$$= \mathbb{E}_X \left(1 - \sum_{l \in \Gamma(X)} p_l(X) + \lambda \sum_{l \in \Gamma(X)} 1\right) \tag{2.21}$$

$$= \mathbb{E}_X \left(1 + \sum_{l \in \Gamma(X)} (\lambda - p_l(X))\right) \ , \tag{2.22}$$

which is minimized by predicting, for all $x \in \mathcal{X}$, all classes l such that $p_l(x) \geq \lambda$.

Thus the Bayes classifier is:

$$\Gamma^*_\lambda(x) = \{l, \mathbb{P}(Y = l|X = x) \geq \lambda\} \tag{2.23}$$

The parameter λ must be set so that on expectation K classes are returned. The expected

number of classes is given by:

$$G(\lambda) \coloneqq \mathbb{E}_X|\Gamma^*_\lambda(X)| = \mathbb{E}_X \sum_{l=1}^{L} \mathbb{1}(l \in \Gamma^*_\lambda(X)) \tag{2.24}$$

$$= \sum_{l=1}^{L} \mathbb{P}_X(l \in \Gamma^*_\lambda(X)) \tag{2.25}$$

$$= \sum_{l=1}^{L} \mathbb{P}_X(p_l(X) \geq \lambda) \ . \tag{2.26}$$

The "right" λ, which we denote by λ^*, is the one that will give a value of G equal to K. The G function is not necessarily invertible, so we will consider the generalized inverse function to obtain the threshold:

$$\lambda^* = G^{-1}(K) \coloneqq \inf\{\lambda, G(\lambda) \leq K\} \ . \tag{2.27}$$

Finally, the expression of the Bayes classifier is:

$$\Gamma^*(x) = \{l, \mathbb{P}(Y = l|X = x) \geq G^{-1}(K)\} \tag{2.28}$$

Estimation.

As in the top-K case, we first need to get an estimate of the conditional probabilities to define our plug-in classifier. This can be achieved by training a deep neural network with the cross-entropy. Once this is done, there is a calibration step. We need to estimate the threshold. Given a calibration set $\{x_1, \ldots, x_n\}$ of unseen samples, we first estimate the G function as follows:

$$\hat{G}(t) = \frac{1}{n} \sum_{i=1}^{n} \sum_{i=1}^{L} \mathbb{1}[\hat{p}_l(x_i) \geq t] \ . \tag{2.29}$$

The estimated threshold is then computed as:

$$\hat{\lambda} \coloneqq \hat{G}^{-1}(K) \ . \tag{2.30}$$

Finally, we can define our plug-in classifier as:

$$\hat{\Gamma}(x) = \{l, \hat{p}_l(x) \geq \hat{G}^{-1}(K)\} \tag{2.31}$$

Notice that for the calibration step (2.29), we only need unlabeled data to estimate the threshold. This is different than conformal prediction for instance, where labeled data is required for the calibration step. This is a huge advantage since unlabeled data is often abundant and cheap whereas labeled data is more precious and it is best to use as much as possible for the training step.

Evaluation.

To evaluate an average-K classifier on a test set $\{(x_1, y_1), \ldots, (x_{n_{test}}, y_{n_{test}})\}$, we can compute the average-K accuracy:

$$\textit{Average-K accuracy} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \mathbb{1}\left[\hat{p}_{y_i}(x_i) \geq \hat{\lambda}\right] \ , \tag{2.32}$$

where $\hat{p}_{y_i}$ represents the estimated conditional probability of the true class.

Notice that the estimation procedure does not guarantee that the average number of classes returned on the test set $\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{L}\mathbb{1}\left[\hat{p}_j(x) \geq \hat{\lambda}\right]$ is equal to K. In practice, when the calibration set and the test set are sufficiently large, this number is very close to K.

2.2.5 Comparison of top-K and average-K classification

Equations (2.12) and (2.28) are quite different. The top-K Bayes classifier simply predicts the top-K largest conditional probabilities for every $x \in \mathcal{X}$. The average-K Bayes classifier predicts all classes whose conditional probabilities are greater a threshold. Therefore, the set size can vary from one example to the other. When a single class has a much higher conditional probability than all of the other ones, the average-K predictor will return a single class, whereas the top-K predictor will return K classes no matter what. On the contrary, when multiple classes are plausible given an image x, the average-K predictor can return a set size greater than K, while the top-K predictor cannot.

When comparing (2.6) and (2.15), it is clear that the average-K Bayes predictor has a lower risk than the top-K Bayes predictor since the constraint is looser. The authors of [LJS21] quantify the difference between the two Bayes classifier. In particular, they provide a lower bound on the gap between the two risks. They conclude that the gain in using average-K over top-K is paricularly important in the case of heterogeneous ambiguity, that is when the level of ambiguity can be very different from on input to another. This is case for Pl@ntNet, as some species are easy to categorize while some species share similarities with many others.

However, it is still relevant to study top-K classification, as top-K accuracy is still a widely reported metric and as it is the most intuitive entry point to set-valued classification. In addition, it is much easier to design a loss function for top-K classification as the constraint applies on a per-sample basis, thus permitting a separable loss (i.e. computable sample by sample). For average-K classification, the inputs are coupled via the global constraint λ which makes it much more difficult to design a loss: given an example $x \in \mathcal{X}$, how many classes should we return ? It depends on the other samples.

2.3 Loss functions

In general, in machine learning, it is not possible to directly minimize the objective we want. Let us consider first the binary case, and say we want to construct a binary classifier $h : \mathcal{X} \to \{-1, 1\}$. To do this, we learn a measurable predictor function $f : \mathcal{X} \to \mathbb{R}$ whose output sign will be the prediction for the target Y. Ultimately, our objective is to minimize the $0/1$ loss so that the classifier makes as little mistake as

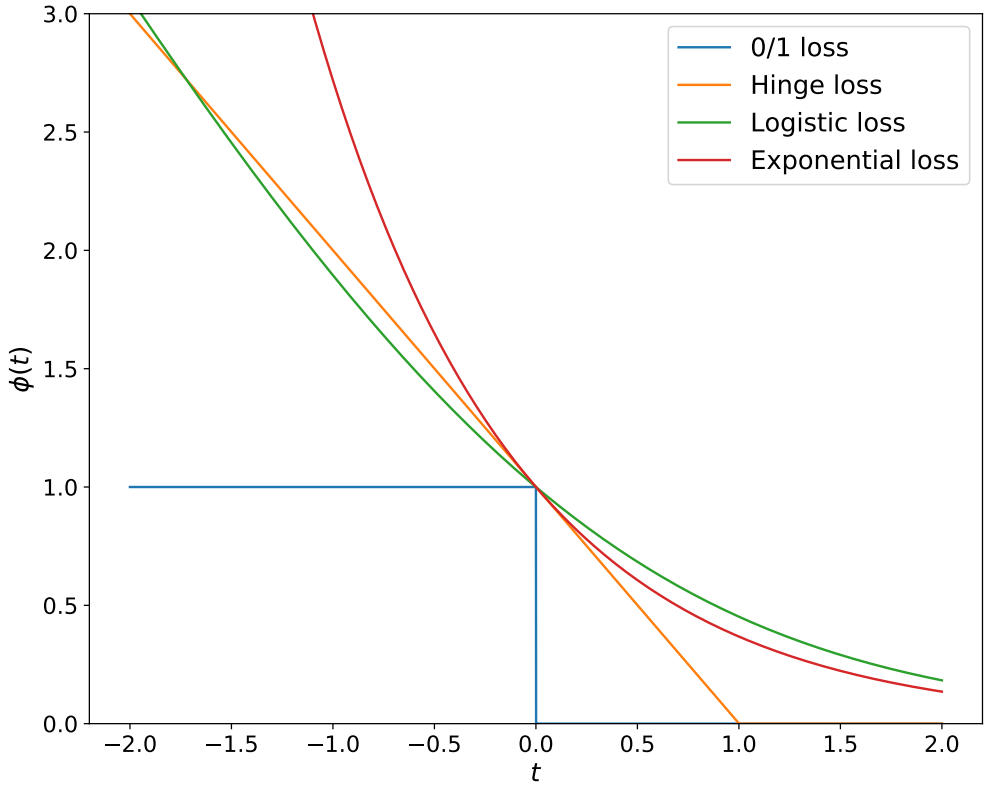


Figure 2.2: *Examples of widespread surrogate losses.*

possible (i.e. get as close as possible to the Bayes classifier):

$$\min_f \mathbb{E}\left(\mathbb{1}[\text{sign}(f(X)) \neq Y]\right) = \mathbb{E}\left(\mathbb{1}[f(X)Y \leq 0]\right) \tag{2.33}$$

$$:= \mathbb{E}\left(\ell(f(X)Y)\right) \quad , \tag{2.34}$$

where sign(α) is equal to one if $\alpha > 0$, -1 otherwise and ℓ denotes the $0/1$ loss.

However, it is not feasible to optimize the $0/1$ loss with traditional algorithms like SGD because it is piecewise constant. Instead, the standard strategy is try to minimize a surrogate loss function $\phi : \mathbb{R} \to \mathbb{R}^+$, an upper bound of the $0/1$ loss. The objective then becomes:

$$\min_f \mathbb{E}\left(\phi(f(X)Y)\right) \quad . \tag{2.35}$$

Some ϕ correspond to famous algorithms like support vector machine [CV95] ($\phi(t) = \max(0, 1-t)$), Adaboost ($\phi(t) = \exp(-t)$) [FS97] or logistic regression ($\phi(t) = \ln(1 + \exp(-t))$)[FHT00]. These examples can be found in Figure 2.2.

A natural and legitimate question arises: if we manage to minimize (2.35), does it mean that we manage to minimize (2.33) as well? This property is called consistency. In the binary case, the answer is yes, if and only if ϕ satisfies a certain condition. To see that, let us write the conditional ϕ risk for a given x:

$$\mathbb{E}[\phi(f(X)Y)|X = x] = p_1(x)\phi(f(x)) + (1 - p_1(x))\phi(-f(x)) \tag{2.36}$$

Let us remove the dependance on x to obtain the generic conditional ϕ-risk:

$$\mathcal{R}_\phi(\alpha, p) = p\phi(\alpha) + (1-p)\phi(-\alpha) \tag{2.37}$$

We say that ϕ is classification calibrated [BJM06; Zha04; LV04] if, for any $p \neq \frac{1}{2}$:

$$\inf_{\alpha, \alpha(2p-1)\leq 0} \mathcal{R}_\phi(\alpha, p) > \inf_{\alpha \in \mathbb{R}} \mathcal{R}_\phi(\alpha, p) \tag{2.38}$$

In other words, the optimal predictor's output has the same sign as $p - \frac{1}{2}$ for all $p \neq \frac{1}{2}$. This means that minimizing the risk will always lead to the best possible decision: the class with the largest probability. This is a necessary and sufficient condition for consistency:

Theorem 2.3.1. (*from [BJM06]*) *ϕ is consistent if and only if ϕ is classification-calibrated.*

In [BJM06], the authors show that when ϕ is convex, then ϕ is classification-calibrated if and only if ϕ is differentiable at 0 and $\phi^{'}(0) < 0$

In the multi-class case [TB07], the conditions to check if a loss is classification-calibrated are much more complex and no such trivial condition exists.

Let us now turn to the case that interests us, that is to say set-valued classification. We can also define the notion of consistency for top-K and average-K classification. A definition was proposed in [Lor20] for plug-in classifiers based on an estimation of the conditional probabilities. We recall it here. Consider an estimator $\hat{\mathbf{p}}(x) \in \Delta^{L-1}$ of the conditional probability vector $\mathbf{p}(x)$, where $\Delta^{L-1} \coloneqq \{\mathbf{p} \in \mathbb{R}^L, \mathbf{p} \geq 0, \sum_{i=1}^L p_i = 1\}$ denotes the simplex. As already explained, we can construct the top-K plug-in classifier $\hat{\Gamma}_{\text{top-K}}(x) = \mathcal{T}_K(\hat{\mathbf{p}}(x))$, and the average-$K$ plug-in classifier $\hat{\Gamma}_{\text{average-K}}(x) = \{l, \hat{p}_l(x) \geq \hat{G}^{-1}(K)\}$. We are interested in the risks of these estimators, respectively $\mathcal{R}(\hat{\Gamma}_{\text{top-K}}) \coloneqq \mathbb{P}(Y \notin \hat{\Gamma}_{\text{top-K}}(X))$ and $\mathcal{R}(\hat{\Gamma}_{\text{average-K}}) \coloneqq \mathbb{P}(Y \notin \hat{\Gamma}_{\text{average-K}}(X))$. As in the binary case, we can not directly minimize these quantities, so instead we minimize a surrogate loss $\ell : \Delta^{L-1} \times [L] \to \mathbb{R}^+$.

Again, the question is: does minimizing this surrogate loss entail minimizing the top-K and average-K risk ? We can also define consistency in the case of top-K and average-K classification [Lor20]:

Definition 2.3.2. We say that a loss is top-K consistent if:

$$\mathcal{R}_\ell(\hat{\mathbf{p}}) \to \mathcal{R}_\ell^* \implies \mathcal{R}(\hat{\Gamma}_{\text{top-K}}) \to \mathcal{R}(\Gamma^*_{\text{top-K}}) \tag{2.39}$$

Definition 2.3.3. We say that a loss is average-K consistent if:

$$\mathcal{R}_\ell(\hat{\mathbf{p}}) \to \mathcal{R}_\ell^* \implies \mathcal{R}(\hat{\Gamma}_{\text{average-K}}) \to \mathcal{R}(\Gamma^*_{\text{average-K}}) \ , \tag{2.40}$$

where $\Gamma^*_{\text{top-K}}$ and $\Gamma^*_{\text{average-K}}$ denote the Bayes top-K and average-K classifier respectively, $\mathcal{R}_\ell(\hat{\mathbf{p}})$ is the ℓ-risk associated to the estimator $\hat{\mathbf{p}}$: $\mathcal{R}_\ell(\hat{\mathbf{p}}) \coloneqq \mathbb{E}(\ell(\hat{\mathbf{p}}(X), Y))$, and $\mathcal{R}_\ell^*$ is the optimal ℓ-risk: $\mathcal{R}_\ell^* \coloneqq \inf_{\hat{\mathbf{p}}} \mathbb{E}(\ell(\hat{\mathbf{p}}(X), Y))$.

In [LJS21], the authors show that strongly proper losses are consistent for both top-K and average-K classification. This emcompasses several classical losses, and in particular cross-entropy. This makes the estimation procedures described in Section 2.2.3 and Section 2.2.4 relevant. In [YK20], the authors propose a consistent top-K loss but we show in Chapter 4 that it performs poorly for deep classification. We propose a modification of this loss by smoothing the top-K operator. We provide a more in-depth review of existing losses for top-K classification in Chapter 4. To the best of our knowledge no loss tailored for average-K classification exists, hence we propose a method in Chapter 5.

Consistency seems like a desirable condition to have for a surrogate loss. It means that if you manage to minimize the ℓ-risk, you manage to minimize the risk. However, this property remains theoretical. In practice, we have a finite, imbalanced training set that we train our model on and try to predict on a test set. Usually, the value of the loss is much smaller for the training set than for the test set. In addition, for deep-learning models the optimization objective is highly non-convex, meaning that we are not guaranteed to find the optimum within our class of functions. Therefore, although consistency can guide the design of loss functions, it does not presume of the actual performance. We will see later that some consistent losses perform worse than others that do not possess this property.

CHAPTER 3

PL@NTNET-300K

Contents

When classifying images, we are faced with two main types of uncertainties[DKD09a]: (i) the *aleatoric* uncertainty that arises from the intrinsic randomness of the underlying process, which is considered irreducible, and (ii) the *epistemic* uncertainty that is caused by a lack of knowledge and is considered to be reducible with additional training data. In modern real-world applications, these two types of uncertainties are particularly difficult to handle. The ever-growing number of classes to distinguish tends to increase the class overlap (and thus the aleatoric uncertainty), and, on the other hand, the long-tailed distribution makes it difficult to learn the less populated classes (and thus increase the epistemic uncertainty). The presence of these two uncertainties is a central motivation for the use of set-valued classifiers, *i.e.*, classifiers returning a set of candidate classes for each image [Chz+21]. Although there are several datasets in the literature that have visually similar classes [NZ08; Maj+13; Yan+15; Rus+15], most of them do not aim to retain both the epistemic and the aleatoric ambiguity present in real-world data.

In this chapter, we propose a dataset designed to remain representative of real-life ambiguity, making it well-suited for the evaluation of set-valued classification methods. This dataset is extracted from real-world images collected as part of the Pl@ntNet project [Aff+17], a large-scale citizen observatory dedicated to the collection of plant occurrences data through image-based plant identification. The key feature of Pl@ntNet is a mobile application that allows citizens to send a picture of a plant they encounter and get a list of the most likely species for that photo in return. The application is used by more than 20 million users in about 170 countries and is one of the main data publishers of GBIF[1], an international platform funded by the governments of many countries around the world to provide free and open access to biodiversity data. Another essential feature of Pl@ntNet is that the training set used to train the classifier is collaboratively enriched and revised. Nowadays, Pl@ntNet covers over 43K species illustrated by nearly 12 million validated images.

The entire Pl@ntNet database would be an ideal candidate for the evaluation of set-valued classification methods. However, it is far too large to allow for widespread use by the machine-learning community. Extracting a subsample from it must be done with care as we want to preserve the uncertainty naturally present in the whole database. The dataset presented in this chapter is constructed by retaining only a subset of the genera of the entire Pl@ntNet database (sampled uniformly at random). All species belonging to the selected genera are then retained. Doing so maintains the original ambiguity as species in the same genus are likely to be visually similar and to share common visual features.

The rest of the chapter is organized as follows. We first introduce the set-valued classification framework in Section 3.1, focusing on two special cases: top-K classification and average-K classification. In Section 3.2, we describe the construction procedure of the Pl@ntNet-300K dataset and show that it contains a large amount of ambiguity. Next, we present in Section 3.3 the metrics of interest for the dataset and propose benchmark results for these metrics, obtained by training several neural networks architectures. In Section 3.4, we compare Pl@ntNet-300K to several existing datasets. In Section 3.5, we

[1] `https://www.gbif.org/`

discuss possible uses of Pl@ntNet-300K. Finally, we provide the link to the dataset in Section 3.6 before concluding.

3.1 Set-valued classification

We adopt the classical statistical setup of multi-class classification. Let L be the number of classes. We denote by $[L]$ the set $\{1, \dots, L\}$ and by $\mathcal{X}$ the input space. Random couples of images and labels $(X, Y) \in \mathcal{X} \times [L]$ are assumed to be generated *i.i.d.* by an unknown joint distribution $\mathbb{P}$. Note that only one label is associated with each image, which differs from the multi-label setting [ZZ14]: P@ntNet-300K is composed of images containing a single specimen of a plant, so there is only one true label per image.

For some plant images, predicting the correct class label (the correct species) does not present much difficulty (consider for instance a common species very distinctive from other species). For other images, however, classifying the photographed specimen with a high degree of confidence is a much harder task, because some species differ only in subtle visual features (see Figure 3.4). In these cases, it is desirable to provide the user with a list of likely species corresponding to the image. We thus need a classifier able to produce sets of classes, also known as a set-valued classifier in the literature [Chz+21].

A set-valued classifier Γ is a function mapping the feature space $\mathcal{X}$ to the set of all subsets of $[L]$ (which we denote by $2^{[L]}$). Using these notations, we thus have $\Gamma : \mathcal{X} \to 2^{[L]}$ instead of $\Gamma : \mathcal{X} \to [L]$ for the classical setting in which the predictor can only predict a single class. Our goal is to build a classifier with low risk, defined as $\mathbb{P}(Y \notin \Gamma(X))$. However, it is not desirable to simply minimize the risk: a set-valued classifier that always returns all classes achieves zero risk but is useless. On the other hand, a classifier is most useful if it returns only the most likely classes given a query image. Therefore, a quantity of interest will be $|\Gamma(x)|$, the number of classes returned by the classifier Γ, given an image $x \in \mathcal{X}$.

In this section, we will examine two optimization problems that lead to different set-valued classifiers. Both of them aim to minimize the risk, but they differ in the way they constrain the set cardinality: either pointwise or on average. For $x \in \mathcal{X}$ and $l \in [L]$, we define the conditional probability $p_l(x) \coloneqq \mathbb{P}(Y = l \mid X = x)$, and estimators of these quantities will be denoted by $\hat{p}_l(x)$. In the following, $K \in [L]$.

The simplest constraint is to require that the number of returned classes is less than K for each input. This results in the following top-K error [LHS15] minimization problem:

$$\begin{aligned} \Gamma^*_{\text{top-K}} \in \arg\min_{\Gamma} \ & \mathbb{P}(Y \notin \Gamma(X)) \\ & \text{s.t. } |\Gamma(x)| \leq K, \ \forall x \in \mathcal{X} \ . \end{aligned} \tag{3.1}$$

The closed form solution to (3.1) exists and is equal to [LHS17]:

$$\Gamma^*_{\text{top-K}}(x) = \mathcal{T}_K(\mathbf{p}(x)) \ . \tag{3.2}$$

Yet, this is not practical since we do not know the distribution $\mathbb{P}$ and thus $p_l(x)$. However, if we have an estimator $\hat{p}_l(x)$ of $p_l(x)$, we can naturally derive the plug-in estimator: $\widehat{\Gamma}_{\text{top-K}} = \text{top}_\text{K}(\hat{\mathbf{p}}(x))$. While the *top-$K$ accuracy* is often reported in benchmarks, only a few works aim to directly optimize this metric [LHS15; LHS16; LHS17; BZK18]. An obvious limitation of top-K classification is that K classes are returned for every data sample, regardless of the difficulty of classifying that sample. Average-K classification allows for more adaptivity. In that setting, the constraint on the size of the predicted set is less restrictive and must be satisfied only on average, leading to:

$$\begin{aligned}\Gamma^*_{\text{average-K}} \in \arg\min_{\Gamma} \ & \mathbb{P}(Y \notin \Gamma(X)) \\ & \text{s.t. } \mathbb{E}_X|\Gamma(X)| \leq K \ .\end{aligned} \tag{3.3}$$

The closed-form solution is derived in [DH17]:

$$\Gamma^*_{\text{average-K}}(x) = \{l \in [L] \ : \ p_l(x) \geq G^{-1}(K)\} \ , \tag{3.4}$$

where the function G is defined as: $\forall t \in [0,1], \quad G(t) = \sum_{l=1}^{L} \mathbb{P}(p_l(X) \geq t)$, and G^{-1} refers to the generalized inverse of G, namely $G^{-1}(u) = \inf\{t \in [0,1] \ : \ G(t) \leq u\}$.

Note that if we define the classifier Γ_t by: $\forall x \in \mathcal{X}$, $\Gamma_t(x) = \{l \in [L], p_l(x) \geq t\}$, then $G(t)$ is the average number of classes returned by Γ_t: $G(t) = \mathbb{E}_X|\Gamma_t(X)|$. From (3.4) we see that the optimal classifier corresponds to a thresholding operation: all classes having a conditional probability greater than $G^{-1}(K)$ are returned, with the threshold chosen so that K classes are returned on average. To compute a plug-in counterpart, we just need to estimate the threshold on a calibration set such that on average on that set, K classes are returned. For technical details, we refer the reader to [DH17].

3.2 Dataset

3.2.1 Label validation and data cleaning

Label validation is based on a weighted majority voting algorithm taking as input the labels proposed by Pl@ntNet users with an adaptive weighting principle according to the user's expertise and commitment. Thus, a single trusted annotator can be enough to validate an image label. On the other hand, images whose labels are proposed by several novice users may not be validated because they do not have sufficient weight. The technical details of this algorithm can be found in the supplementary material. At the time of the construction of Pl@ntNet-300K, the total number of annotators in the Pl@ntNet database was equal to 2,079,003. The average number of annotators per image is equal to 2.03.

In addition to the label validation procedure, Pl@ntNet's pipeline includes other data cleaning procedures: (i) automated filtering of inappropriate or irrelevant content (faces, humans, animals, buildings, etc.) using a CNN and user reports, and (ii) filtering on image quality (evaluated by users).

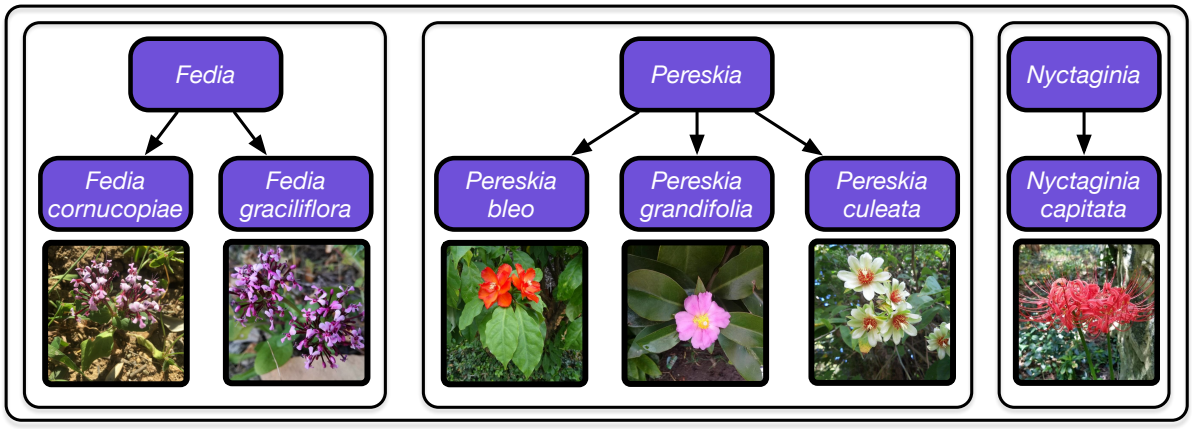


Figure 3.1: *Genus taxonomy: we display three genera present in the proposed dataset—Fedia, Pereskia and Nyctaginia—which contain respectively two, three and one species.*

3.2.2 Construction of Pl@ntNet-300K

In taxonomy, species are organized into genera, with each genus containing one or more species, and the different genera do not overlap, as illustrated in Figure 3.1.

Instead of retaining randomly selected species or images from the entire Pl@ntNet dataset, we choose to retain randomly selected genera and keep all species belonging to these genera. This choice aims to preserve the large amount of ambiguity present in the original database, as species belonging to the same genus tend to share visual features. The dataset presented in this chapter is constructed by sampling uniformly at random only 10% of the genera of the whole Pl@ntNet database.

We then retain only species with more than 4 images, resulting in a total of 303 genera and $L = 1\,081$ species. The images are divided into a training set, a validation set and a test set.[2] For each species, 80% of the images are placed in the training set ($n_{train} = 243\,916$), 10% in the validation set ($n_{val} = 31\,118$), and 10% in the test set ($n_{test} = 31\,112$), with at least one image of each species in each set. More formally, given a class j containing n_j images, $n_{val,j} = \lceil 0.1 \times n_j \rceil$, $n_{test,j} = \lceil 0.1 \times n_j \rceil$ and $n_{train,j} = n_j - n_{val,j} - n_{test,j}$. This represents a total of $n_{tot} = n_{train} + n_{val} + n_{test} = 306\,146$ color images. The average image size is (570, 570, 3), ranging from (180, 180, 3) to (900, 900, 3). The construction of the dataset preserves the class imbalance. To show this, we plot the Lorenz curves [Gas71] of the entire Pl@ntNet dataset and that of the Pl@ntNet-300K dataset in Figure 3.2.

3.2.3 Epistemic (model) uncertainty

Epistemic uncertainty refers mainly to the lack of data necessary to properly estimate the conditional probabilities. In Pl@ntNet, the most common species are easily observed by users in the wild and therefore represent a large fraction of the images, while the rarest species are harder to find and therefore less frequent in the database. In Figure 3.2, we see that 80% of the species (the ones with the lowest number of images) account

[2]The division is performed at the species level due to the long-tailed distribution.

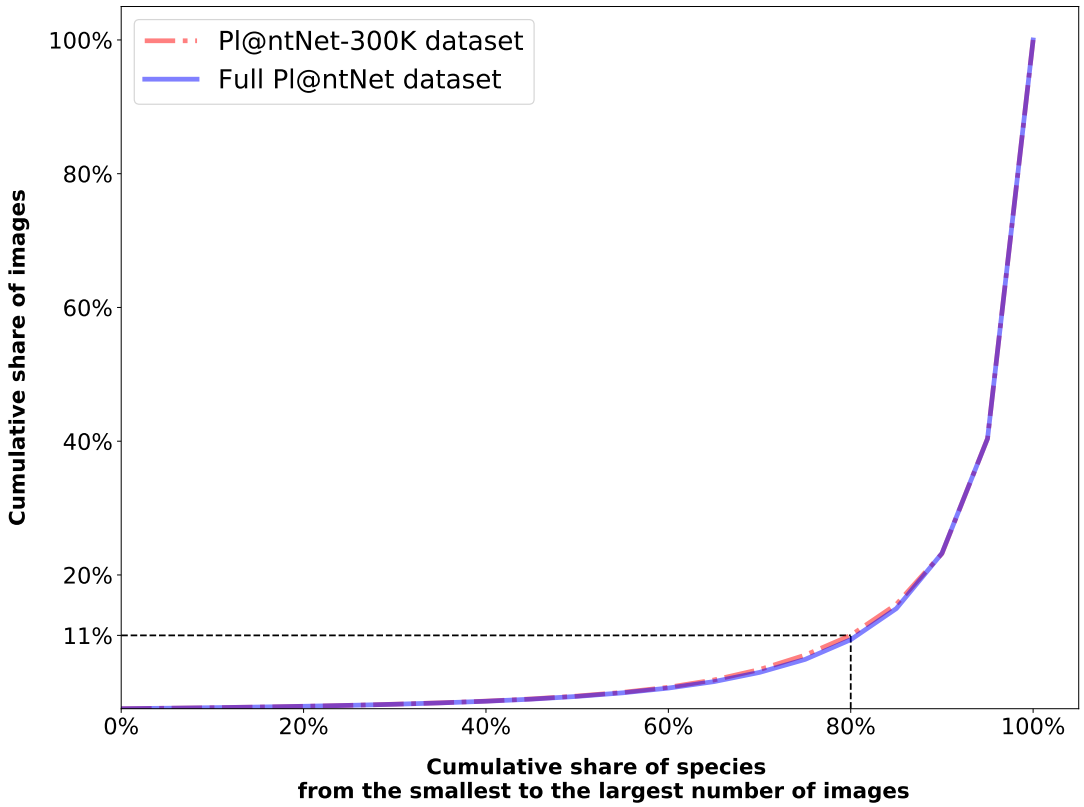


Figure 3.2: *Lorenz curves of the Pl@ntNet database and the proposed dataset. Note that, for fair comparisons, we discard species with less than 4 images in the Pl@ntNet database.*

for only 11% of the total number of images. Hence, training machine learning models is challenging for such a dataset, since for many classes the model only has a handful of images to train on, making identification difficult for these species.

In addition to the long-tailed distribution issue, epistemic uncertainty also arises from the high intra-species variability. Plants may take on different appearances depending on the season (*e.g.*, , flowering time). Furthermore, a user of the application may photograph only a part of the plant (for instance, the trunk and not the leaves). As a last example, flowers belonging to the same species can have different colors. Figure 3.3 shows some examples of these phenomena which contribute to high intra-class variability, making it more challenging to model the species.

3.2.4 Aleatoric (data) uncertainty

In our case, the source of aleatoric uncertainty mostly resides in the limited information we are given to make a decision (assign a label to a plant). Some species, especially those belonging to the same genus, can be visually very similar. For example, consider the case where two species produce the same flowers but different leaves, typically because they have evolved differently from the same parent species. If a person photographs only the flower of a specimen of one of the two species, then it will be impossible, even for an expert, to know which species the flower belongs to. The discriminative information is not present in the image.

Guizotia abyssinica | *Diascia rigescens* | *Lapageria rosea* | *Casuarina cunninghamiana* | *Freesia alba*

Figure 3.3: *Examples of visually different images belonging to the same class.*

The combination of this irreducible ambiguity with images of non-optimal quality (non-adapted close-up, low-light conditions, etc.) results in pairs of images that belong to different species but are difficult or even impossible to distinguish, see Figure 3.4 for illustration. In this figure, we show the ambiguity between pairs of species, but we could find similar examples involving a larger number of species. Thus, even an expert botanist might fail to assign a label to such pictures with certainty. This is embodied by $p_l(x)$: given an image, multiple classes are possible.

3.3 Evaluation

3.3.1 Metric

We consider two main metrics to evaluate set valued predictors on Pl@ntNet-300K: *top-K accuracy* and *average-K accuracy*. Let S denote a set of n (input, label) pairs: $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$.

Top-K accuracy [LHS16] is a widely used metric often reported in benchmarks. *Average-K accuracy* is a much less common metric that derives from average-K classification [DH17]:

$$\textit{average-K accuracy}(S) = \frac{1}{n} \sum_{(x_i,y_i)\in S} \mathbb{1}_{[y_i \in \widehat{\Gamma}_{\text{average-K}}(x_i)]} \text{ s.t. } \frac{1}{n} \sum_{(x_i,y_i)\in S} |\widehat{\Gamma}_{\text{average-K}}(x_i)| \leq K \ , \tag{3.5}$$

where $\widehat{\Gamma}_{\text{average-K}}$ is a set-valued classifier constructed using the training data.

For Pl@ntNet-300K, both *top-K accuracy* and *average-K accuracy* mainly reflect the performance of the set-valued classifier on the few classes which represent most of the images. If we wish to capture the ability of a set-valued classifier to return relevant set of species for all classes, we will examine *macro-average top-K accuracy* and *macro-average*

Figure 3.4: *Examples of visually similar images belonging to two different classes.*

average-K accuracy which simply consist in computing respectively *top-K accuracy* and *average-K accuracy* for each class, and then computing the average over classes. For *macro-average average-K accuracy*, the constraint on the average size of the set must hold for the entire set S.

To derive both classifiers, one can first obtain an estimate of the conditional probabilities $\hat{p}_l(x)$ and then derive the plug-in classifiers, as explained in Section 3.1. Our hope is for the Pl@ntNet-300K dataset to encourage novel ways to derive the set-valued classifiers $\widehat{\Gamma}_{\text{top-k}}$ and $\widehat{\Gamma}_{\text{average-K}}$ to optimize respectively the *top-K accuracy* and the *average-K accuracy*. Notice that a few works already propose methods to optimize the *top-K accuracy* [LHS15; LHS16; LHS17; BZK18].

3.3.2 Baseline

This section provides a baseline evaluation of the plug-in classifiers. We train several deep neural networks with the cross-entropy loss: ResNets [He+16], DenseNets, [Hua+17], InceptionResNet-v2 [Sze+17], MobileNetV2 [San+18], MobileNetV3 [How+19], EfficientNets [TL19], Wide ResNets [ZK16], AlexNet [KSH12], Inception-v3 [Sze+16], Inception-v4 [Sze+17], ShuffleNet [Zha+18], SqueezeNet [Ian+16], VGG [SZ15] and Vision Transformer [Dos+21].

All models are pre-trained on ImageNet. During training, images are resized to 256 and a random crop of size 224×224 is extracted. During test time, we take the centered crop.

The models are optimized with SGD with a momentum of 0.9 with the Nesterov acceleration [Rud16a]. We use a batch size of 32 for all models and a weight decay of 1.10^{-4}. The number of epochs, initial learning rate and learning rate schedule used for each model can be found in the supplementary material. For the plug-in classifier $\widehat{\Gamma}_{\text{average-K, plug-in}}$, we compute the threshold λ_{val} on the validation set and use that same

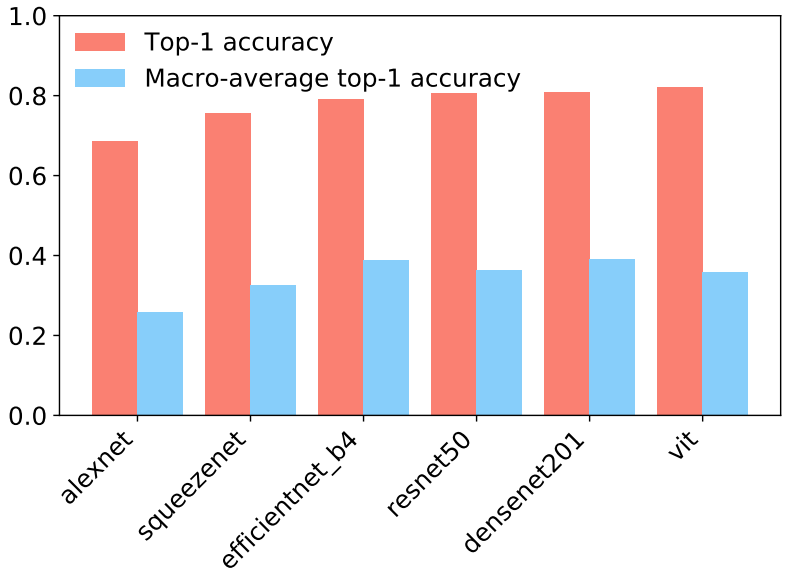


Figure 3.5: *Pl@ntNet-300K test top-1 accuracy and macro-average top-1 accuracy for several neural networks.*

Number of images	Mean bin accuracy
$0 - 10$	0.09
$10 - 50$	0.35
$50 - 500$	0.59
$500 - 2000$	0.79
> 2000	0.93

Table 3.1: *Test accuracy depending on the number of images per class in the training set. Obtained with a ResNet50.*

threshold to compute the *average-K accuracy* on the test set.

3.3.3 Difficulty of Pl@ntNet-300K

Figure 3.5 highlights the significant gap between Pl@ntNet-300K *top-1 accuracy* and *macro-average top-1 accuracy*. This is a consequence of the long-tailed distribution: the few classes that represent most of the images are easily identified, which results in high *top-1 accuracy*. However, this seemingly high *top-1 accuracy* is misleading, as models struggle with classes with few images (which are a majority, see Figure 3.2). This effect is illustrated in Table 3.1, which shows that the *top-1 accuracy* depends strongly on the number of images in the class.

Figure 3.8(a) shows the correlation between Pl@ntNet-300K *macro-average top-1 accuracy* and ImageNet *macro-average top-1 accuracy* (note that as the ImageNet test set is balanced, *top-1 accuracy* and *macro-average top-1 accuracy* coincide). As expected, the two metrics are positively correlated: deep networks allow to model complex features.

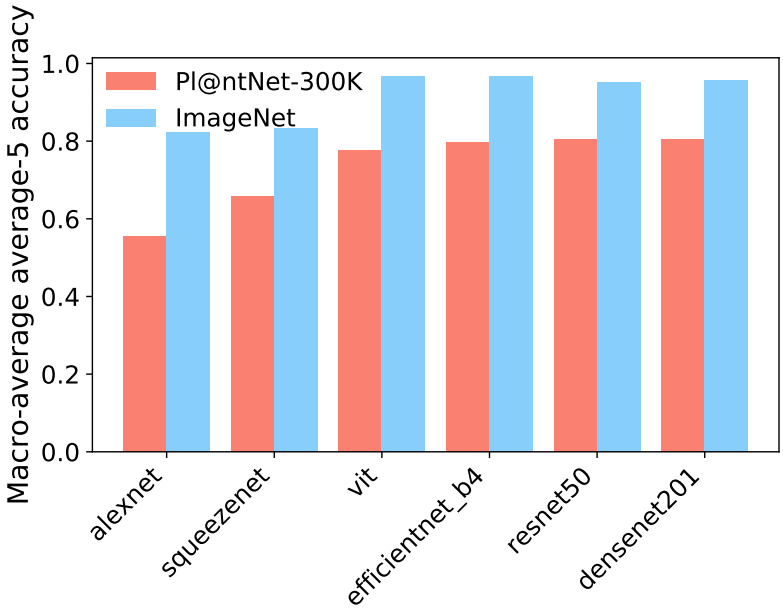


Figure 3.6: *Pl@ntNet-300K vs ImageNet macro-average average-5 accuracy for several models (evaluated on the test set).*

It works well both on ImageNet and Pl@ntNet-300K. Interestingly, due to the difference between the two datasets (long-tailed distribution, class ambiguity, ...), some models which perform similarly on ImageNet yield very different on Pl@ntNet-300K (inception_v3, densenet201), and vice versa.

In Figure 3.8(a) we can notice that ImageNet *macro-average top-1 accuracy* and Pl@ntNet-300K *macro-average top-1 accuracy* vary at different scales: the former goes up to 80% while the latter does not exceed 40%, making Pl@ntNet-300K a challenging dataset with both epistemic and aleatoric uncertainty at play.

This can can also be seen in Figure 3.6: some models reach a *macro-average average-5 accuracy* of 97% for ImageNet, while that metric does not exceed 80% for Pl@ntNet-300K, which suggests that progress could be made with appropriate learning strategies.

To support that claim, we asked a botanist to label a mini dataset extracted from the Pl@ntNet-300K test set. The dataset is constructed as follows: we extract all species from two groups (*Crotalaria* and *Lupinus*), and select at most 5 images per species (randomly sampled). This results in 83 images. The botanist was asked to provide a set of possible species for each image. This results in an error rate of 20.5% for an average of 4,1 species returned (ranging from 1 to 10). We compare the botanist performance with that of several neural networks by calibrating the conditional probabilities' threshold to obtain on average 4,1 species on the mini-dataset. The results are reported in Figure 3.7, and show that the gap between the botanist error rate and the best-performing model is large (from 0.2 to 0.33), which suggests that there is room for improvement in the performance of average-K classifiers.

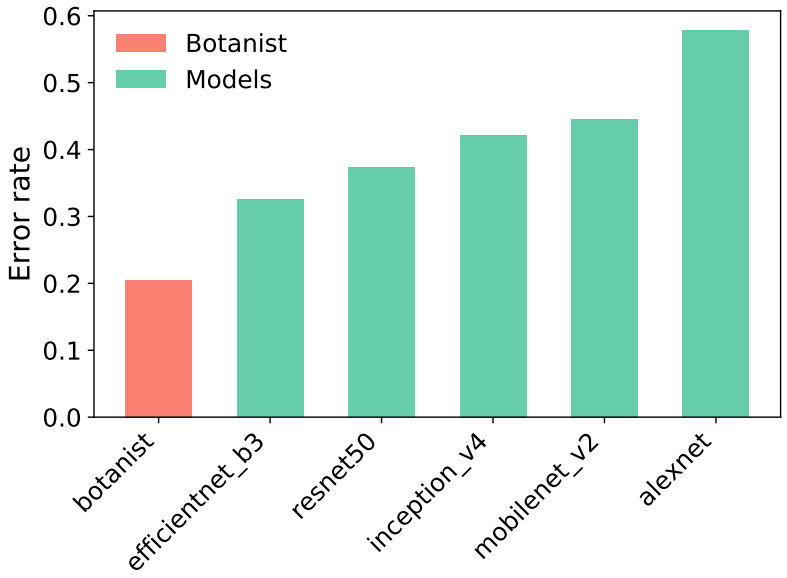


Figure 3.7: *Error rate on the mini test set of an expert compared to several neural networks. All models and the expert return on average 4.1 species on the mini test set.*

3.3.4 Top-K vs Average-K

From Equation (3.1) and (3.3), it is clear that the Bayes average-K classifier has a lower risk than the Bayes top-K classifier. Therefore, a model that accurately estimates the conditional probabilities should yield a better *average-K accuracy* than *top-K accuracy*. This is what can be observed in Figure 3.8(b) which shows the correlation between *macro-average top-5 accuracy* and *macro-average average-5 accuracy*. As expected, the two metrics are positively correlated. However, the relationship does not appear to be trivial and Figure 3.8(b) shows models with similar *macro-average average-5 accuracies* having very different *macro-average top-5 accuracy* and vice versa. For an in-depth comparison of average-K classification and top-K classification, we refer the reader to [Lor20]. Both metrics are of their own interest and deserve a specific treatment as they capture two different settings: *top-K accuracy* evaluates the performance of a classifier that systematically returns K classes, while *average-K accuracy* evaluates the performance of a classifier that returns sets of varying size (depending on the input), with the constraint to return K classes on average.

3.3.5 Evaluation of existing set-valued classification methods

To the best of our knowledge, there is no existing loss designed to specifically optimize *average-K accuracy*. For top-K classification, the most recent loss designed to optimize *top-K accuracy* is the one by [BZK18]. We report the *top-5 accuracy* obtained by training this loss with $K = 5$ on Pl@ntNet-300K in the supplementary material. The results are close to what is obtained with the cross entropy loss. However, this topic is still open research and our hope in releasing Pl@ntNet-300K is precisely to encourage novel methods for optimizing such metrics.

3.4 Related work

Fined-Grained Visual Categorization (FGVC) is about discriminating visually similar classes. In order to better learn fine-grained classes, several approaches have been proposed by the FGVC community, including multi-stage metric learning [Qia+15], high-order feature interaction [LRM15; Cui+17], and different network architectures [FZM17; Ge+16]. However, these approaches focus on optimizing *top-1 accuracy*. Set-valued classification, on the other hand, consists in returning more than a single class to reduce the error rate, with a constraint on the number of classes returned. Therefore, FGVC and set-valued classification methods are not mutually exclusive but rather complementary.

Several FGVC datasets, which exhibit visually similar classes, have been made publicly available by the community. They cover a variety of domains: aircraft [Maj+13], cars (Compcars [Yan+15], Census cars [Geb+17]), birds (CUB200 [Wel+10]), flowers (Oxford flower dataset [NZ08]).

However, most of these datasets focus exclusively on proposing visually similar classes (aleatoric uncertainty) with a limited amount of epistemic uncertainty. This is the case for balanced datasets which have approximately the same number of images per class, or with small intra-class variability such as aircraft and cars datasets, where most examples within a class are nearly the same except for angle, lighting, etc.

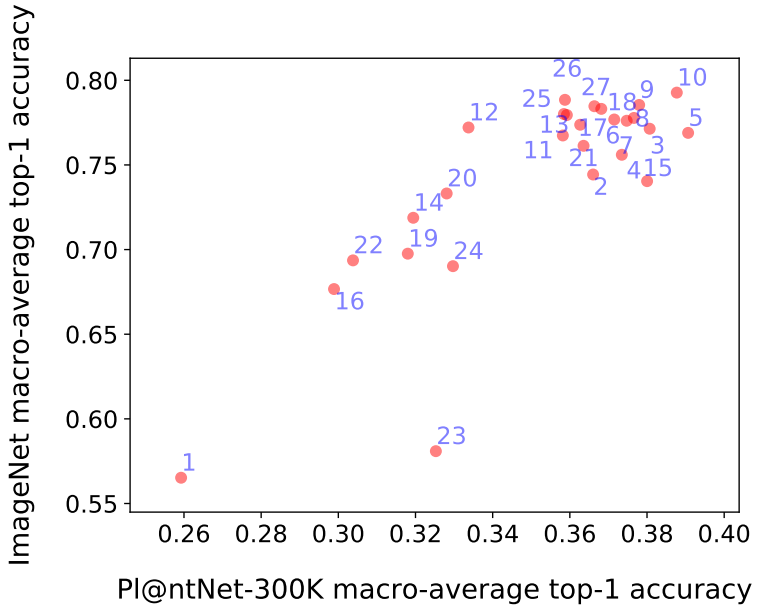
ImageNet [Rus+15] has several visually similar classes, organized in groups: it contains many bird species and dog breeds. However, these groups of classes are very different: dogs, vehicles, electronic devices, etc. Besides, ImageNet does not exhibit a strong class imbalance.

Several of these datasets were constructed by web-scraping, which can be prone to noisy labels and low-quality images.

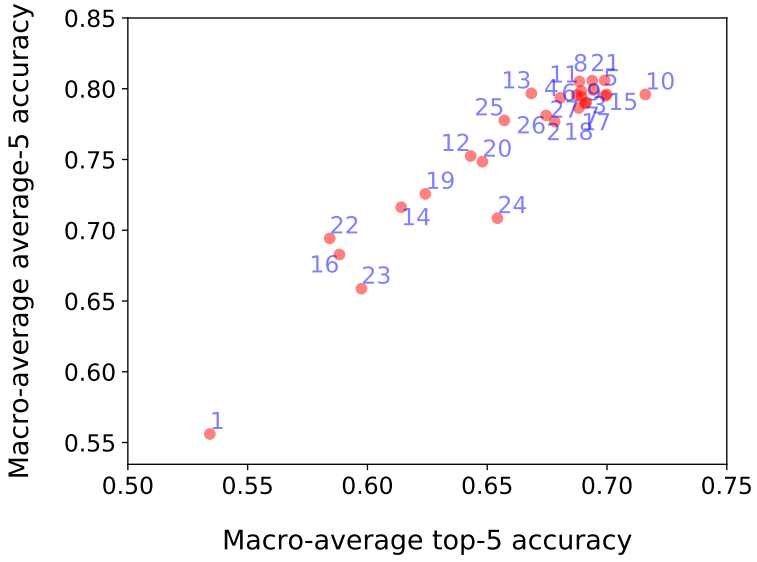
Most similar to our dataset is the iNat2017 dataset [Hor+18]. It contains images from the citizen science website iNaturalist. The images, posted by naturalists, are validated by multiple citizen scientists. The iNat2017 dataset contains over 5000 classes that are highly unbalanced. However, iNat2017 does not only focus on plants but proposes several other 'super-classes' such as *Fungi*, *Reptilia*, *Insecta*, etc. Moreover, the authors selected all classes with a number of observations greater than 20, whereas we choose to randomly sample 10% of the genera of the entire Pl@ntNet database and keep all species belonging to these groups with a number of observations greater than 4. We argue that keeping all species of the same genus maximizes aleatoric uncertainty because species belonging to the same genus tend to share visual features.

Finally, a plant disease dataset is introduced in [Sla+16], containing 4483 images downloaded from the web spread across 15 classes. This is a very different scale than Pl@ntNet-300K. We summarize the properties of the mentioned datasets in Table 3.2.

[3]The architectures chosen are: alexnet (1), densenet121 (2), densenet161 (3), densenet169 (4), densenet201 (5), efficientnet_b1 (6), efficientnet_b1 (7), efficientnet_b2 (8), efficientnet_b3 (9), efficientnet_b4 (10), inception_resnet_v2 (11), inception_v3 (12), inception_v4 (13), mobilenet_v2 (14), mobilenet_v3_large (15), mobilenet_v3_small (16), resnet101 (17), resnet152 (18), resnet18 (19), resnet34 (20), resnet50 (21), shufflenet (22), squeezenet (23), vgg11 (24), vit_base_patch16_224 (25),



(*a*) *ImageNet macro-average top-1 accuracy vs. Pl@ntNet-300K macro-average top-1 accuracy (evaluated on the test set).*



(*b*) *Pl@ntNet-300K macro-average average-5 accuracy vs. macro-average top-5 accuracy (evaluated on the test set).*

Figure 3.8: *Benchmark for several popular deep neural network architectures*[3].

3.5 Possible uses of Pl@ntNet-300K

Although we are convinced of the need to design new set-valued methods due to the ever-increasing amount of classes to discriminate, the properties of Pl@ntNet-300K described in Section 3.2 make it an ideal candidate for various other tasks. The strong class imbalance can be used by researchers to evaluate new algorithms specifically

wide_resnet101_2 (26), wide_resnet50_2 (27).

Table 3.2: *Comparison of several datasets with Pl@ntNet-300K. "Focused domain" indicates whether the dataset is made up of a single category* (i.e., *cars*) *and "Ambiguity preserving sampling" indicates whether in the construction of the dataset, all classes belonging to the same parent in the class hierarchy were kept or not* (*in our case, the parent corresponds to the genus level*).

	Human-in-the-loop labeling	Long-tailed distribution	Intra-class variability	Focused domain	Ambiguity preserving sampling
Plant disease dataset	✗	✗	✗	✓	✗
CUB200	✗	✗	✗	✓	✗
Oxford flower dataset	✗	✗	✓	✓	✗
Aircraft dataset	✓	✗	✗	✓	✗
Compcars	✗	✗	✗	✓	✓
Census cars	✗	✗	✗	✓	✓
ImageNet	✗	✗	✓	✗	✗
iNat2017	✓	✓	✓	✗	✗
Pl@ntNet-300K	✓	✓	✓	✓	✓

designed for tackling class imbalance [Zho+20; Cao+19]. Pl@ntNet-300K contains a large amount of aleatoric uncertainty resulting from many visually similar classes. It can therefore be used as an FGVC dataset to evaluate methods that aim to optimize *top*-1 *accuracy* for such datasets, see for instance [LRM15; FZM17; Cui+17]. Finally, in this chapter, we do not use the genus information and thus do not exploit the hierarchical structure of the problem. In this sense, we adopt the flat classification approach described in [SF11a]. This is consistent with ImageNet [Rus+15] or CIFAR-100 [Kri09], where a hierarchy does exist but is rarely used in benchmarks. However, researchers are free to use the genus information to evaluate hierarchical classification methods on Pl@ntNet-300K.

3.6 Data access and additional resources

The Pl@ntNet-300K dataset can be found here:

`https://doi.org/10.5281/zenodo.5645731`.

It is organized in three folders named "`train`", "`val`" and "`test`". Each of these folders contains $L = 1\,081$ subfolders. We provide the correspondence between the names of the subfolders and the names of the classes in the file "`plantnet300K_species_id_2_name.json`". We also provide a metadata file named "`plantnet300K_metadata.json`" containing for each image the following information: the species identifier (class), the organ of the plant (flower, leaf, bark, ...), the author's name, the license and the split (*i.e.,* train, validation or test set). A GitHub repository containing the code to reproduce the experiments of this chapter (where potential issues related to the dataset can be reported too) is available at: `https://github.com/plantnet/PlantNet-300K/`. To date (2023), the dataset has

been viewed 20,000 times and downloaded 5,000 times.[4]

3.7 Conclusion

In this chapter, we share and discuss a novel plant image dataset, called Pl@ntNet-300K, obtained as a subset of the entire Pl@ntNet database and intended primarily for evaluating set-valued classification methods. Unlike previous datasets, Pl@ntNet-300K is designed to preserve the high level of ambiguity across classes of the initial real-world dataset as well as its long-tailed distribution. To evaluate set-valued predictors on Pl@ntNet-300K, we investigate two different metrics: *macro-average top-K accuracy* and *macro-average average-K accuracy*, which is a more challenging task requiring to predict sets of various size but still equal to K on average. Our results suggest that there is room for new set-valued prediction methods that would improve the performance of average-K classifiers. We hope that Pl@ntNet-300K can serve as a reference dataset for this problem, which is our main motivation for releasing and sharing it with the community. We also stress that Pl@ntNet-300K can also be used to evaluate new methods for long-tailed classification and FGVC. In the next chapters, we propose methods to optimize respectively top-K and average-K accuracy use Pl@ntNet-300K to test our ideas.

[4]Source: `https://doi.org/10.5281/zenodo.5645731`

3.8 Appendix

3.8.1 Pl@ntNet label validation process

Only Pl@ntNet observations with a valid species name were included in the Pl@ntNet-300K dataset. The species name validation process of Pl@ntNet is based on a weighted majority voting algorithm taking as input the labels proposed by Pl@ntNet users with a principle of adaptive weights depending on the user's expertise and engagement. More precisely, the most probable label y_i of an image x_i is computed as:

$$y_i = \underset{k \in 1,\dots,d}{\arg\max} \sum_{u \in U_i} w_u \mathbb{1}(y_i^u = k) \ ,$$

where U_i is the set of users who suggested a label for the image x_i and y_i^u is the label proposed by the user u. The weight w_u of a user u is computed as:

$$w_u = n_u^\alpha - n_u^\beta + b_0 \ ,$$

where n_u is the number of species observed by user u (*i.e.*, the number of species for which the user is the author of at least one valid image), plus the number of distinct labels y_i^u proposed by a user u in the whole dataset. The constant power values $\alpha = 0.5$, $\beta = 0.2$ and b_0 are determined empirically.
To be considered as valid, the label y_i of an image x_i must satisfy:

$$\sum_{u \in U_i} w_u \mathbb{1}(y_i^u = y_i) > \theta$$

where θ is a fixed threshold. Images with non-valid labels were discarded from Pl@ntNet-300K dataset.

3.8.2 Hyperparameters

The hyperparameters used for the experiments in Section 4 of the paper can be found in Table 3.3.

Table 3.3: *Learning rate, number of epochs and learning rate schedule for the different models. At each learning rate decay, the learning rate is divided by ten.*

Models	Initial learning rate	Number of epochs	First decay	Second decay
mobilenet_v2, mobilenet_v3_large, resnet 18, 34, 50, 101, 152, densenet 121, 161, 169, 201, inception_v3, inception_v4, inception_resnet_v2, wide_resnet50_2, wide_resnet101_2, shufflenet	0.01	30	20	25
vgg11, alexnet, mobilenet_v3_small, squeezenet	0.001	30	20	25
efficientnet b0, b1, b2, b3, b4	0.01	20	10	15
vit	5e-4	20	15	-

3.8.3 Evaluation of existing set-valued classification methods

Figure 3.9 compares the top-5 accuracy of several models when they are trained with either the cross entropy loss or the top-K loss by [BZK18]. The hyperparameters used for training the top-K loss are the same as in Section 4. The smoothing parameter τ of the top-K loss is set to 1.0.

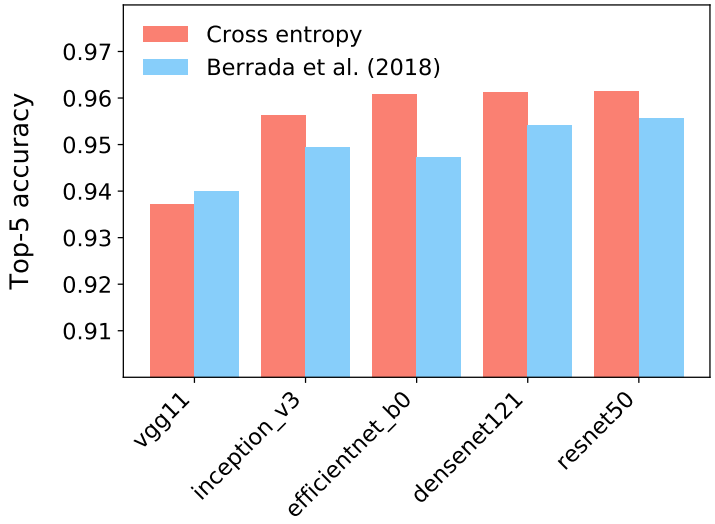


Figure 3.9: *Comparison of top-5 accuracy of models trained with either the cross entropy loss or the loss by [BZK18] (with $K = 5$)*

CHAPTER 4

TOP-K LOSS FOR DEEP LEARNING

Contents

4.1 Introduction

Fine-grained visual categorization (FGVC) has recently attracted a lot of attention [Wan+22], in particular in the biodiversity domain [Hor+18; Gar+21; VH+15]. In FGVC, one aims to classify an image into subordinate categories (such as plant or bird species) that contain many visually similar instances. The intrinsic ambiguity among the labels makes it difficult to obtain high levels of top-1 accuracy as is typically the case with standard datasets such as CIFAR10 [Kri09] or MNIST [LeC+98]. For systems like Merlin [VH+15] or Pl@ntNet [Aff+17], due to the difficulty of the task, it is generally relevant to provide the user with a set of classes in the hope that the true class belongs to that set. In practical applications, the display limit of the device only allows to give a few labels back to the user. A straightforward strategy consists in returning a set of K classes for each input, where K is a small integer with respect to the total number of classes. Such classifiers are called top-K classifiers, and their performance is evaluated with the well known top-K accuracy [LHS15; Rus+15]. While such a metric is very popular for evaluating applications, common learning strategies typically consist in learning a deep neural network with the cross-entropy loss, neglecting the top-K constraint in the learning step.

Yet, recent works have focused on optimizing the top-K accuracy directly. [LHS15] have introduced the top-K hinge loss and a convex upper-bound, following techniques introduced by [UBG09] for ranking. A limit of this approach was raised by [BZK18], as they have shown that the top-K hinge loss by [LHS15] can not be directly used for training a deep neural network. The main arguments put forward by the authors to explain this practical limitation are: (i) the non-smoothness of the top-K hinge loss and (ii), the sparsity of its gradient. Consequently, they propose a smoothed alternative adjustable with a temperature parameter. However, their smoothing procedure is computationally costly when K increases (as demonstrated in our experiments), despite the efficient algorithm they provide to cope with the combinatorial nature of the loss. Moreover, this approach has the drawback to be specific to the top-K hinge loss introduced by [LHS15].

In contrast, we propose a new top-K loss that relies on the smoothing of the top-K operator (the operator returning the K-th largest value of a vector). The smoothing framework we consider, the perturbed optimizers [Ber+20], can be used to smooth variants of the top-K hinge loss but could independently be considered for other learning tasks such as K-nearest neighbors or top-K recommendation [HWC19; CAS16]. Additionally, we introduce a simple variant of our loss to deal with imbalanced datasets. Indeed, for many real-world applications, a long-tailed phenomenon appears [Ree01]: a few labels enjoy a lot of items (*e.g.,* images), while the vast majority of the labels receive only a few items, see for instance a dataset like Pl@ntnet-300k [Gar+21] for a more quantitative overview. We find that the loss by [BZK18] fails to provide satisfactory results on the tail classes in our experiments. On the contrary, our proposed loss based on uneven margins outperforms the loss from [BZK18] and the LDAM loss [Cao+19], a loss designed for imbalance cases known for its very good performance in fine-grained visual classification challenges [Wan+21]. To the best of our knowledge, our proposed loss is the first loss function tackling both the top-K classification and

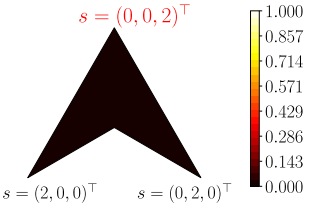
class imbalance problems jointly.

4.2 Related work

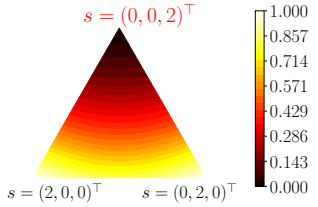
Several top-K losses have been introduced and experimented with in [LHS15; LHS16; LHS17]. However, the authors assume that the inputs are features extracted from a deep neural network and optimize their losses with SDCA [SSZ13]. [BZK18] have shown that the top-K hinge loss from [LHS15] could not be directly used in a deep learning optimization pipeline. Instead, we are interested in end-to-end deep neural network learning. The state-of-the art top-K loss for deep learning is that of [BZK18], which is a smoothing of a top-K hinge loss by [LHS15]. The principle of the top-K loss of [BZK18] is based on the rewriting the top-K hinge loss of [LHS15] as a difference of two maxes on a combinatorial number of terms, smooth the max with the logsumexp, and use a divide-and-conquer approach to make their loss tractable. Instead, our approach relies on smoothing the top-K operator and using this smoothed top-K operator on a top-K calibrated loss recently proposed by [YK20]. Our approach could be used out-of-the box with other top-K hinge losses. In contrast, the smoothing method of [BZK18] is tailored for the top-K hinge loss of [LHS15], which is shown to be not top-K calibrated in [YK20].

For a general theory of smoothing in optimization, we refer the reader to [BT12; Nes05] while for details on perturbed optimizers, we refer the reader to [Ber+20] and references therein. In the literature, other alternatives have been proposed to perform top-K smoothing. [Xie+20] formulate the smooth top-K operator as the solution of a regularized optimal transport problem between well-chosen discrete measures. The authors rely on a costly optimization procedure to compute the optimal plan. [XE19] propose a smoothing of the top-K operator through K successive *softmax*. Besides the additional cost with large K, the computation of K successive *softmax* brings numerical instabilities.

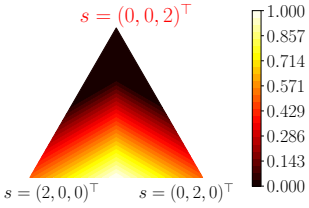
Concerning imbalanced datasets, several recent contributions have focused on architecture design [Zho+20; Wan+21]. Instead, we focus here on the design of the loss function and leverage existing popular neural networks architectures. A popular loss for imbalanced classification is the focal loss [Lin+17] which is a modification of the cross entropy where well classified-examples induce a smaller loss, putting emphasis on difficult examples. Instead, we use uneven margins in our formulation, requiring examples of the rarest classes to be well classified by a larger margin than examples of the most common classes. Uneven margin losses have been studied in the binary case in [Sco12; LST03; IMV19]. For the multi-class setting, the LDAM loss [Cao+19] is a widely used uneven margin loss which can be seen as a cross entropy incorporating uneven margins in the logits. Instead, our imbalanced top-K loss relies on the smoothing of the top-K operator.



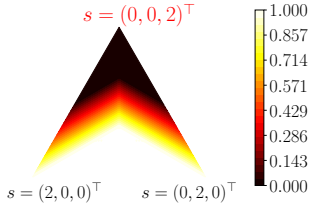
(a) *Top-K:* $\ell = \ell^K$.



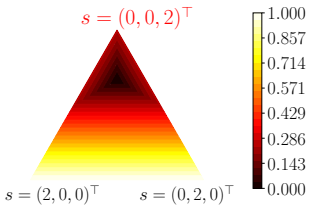
(b) *Cross-entropy:* $\ell = \ell_{CE}$.



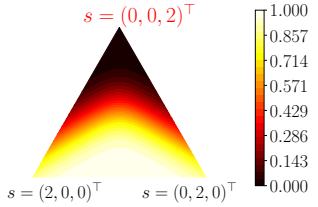
(c) *Multi-class hinge:* $\ell = \ell^K_{Hinge}$.



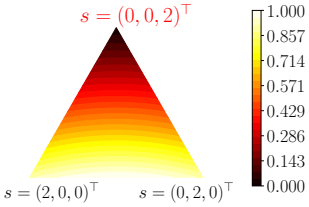
(d) *Calibrated hinge:* $\ell = \ell^K_{Cal.\ Hinge}$.



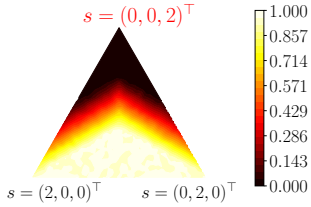
(e) *Convexified hinge:* $\ell = \ell^K_{CVXHinge}$.



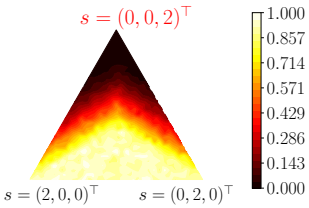
(f) *Smoothed hinge* $\ell^{K,0.1}_{Smoothed\ Hinge}$.



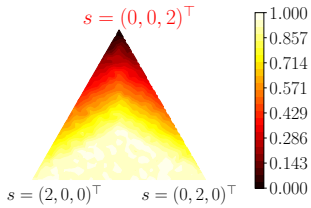
(g) *Smoothed hinge* $\ell^{K,1}_{Smoothed\ Hinge}$.



(h) *Noised balanced:* $\ell^{K,0.3,30}_{Noised\ bal.}$.



(i) *Noised balanced:* $\ell^{K,1,30}_{Noised\ bal.}$.



(j) *Noised imbalanced:* $\ell^{K,1,30,5}_{Noised\ Imbal.}$.

Figure 4.1: *Level sets of the function* $\mathbf{s} \mapsto \ell(\mathbf{s}, y)$ *for different losses described in Table 4.1, for* $L = 3$ *classes,* $K = 2$ *and a true label* $y = 3$ *(corresponding to the upper corner of the triangles). For visualization the loss are rescaled between 0 and 1, and the level sets are restricted to vector* $\mathbf{s} \in 2 \cdot \Delta^2$. *The losses have been harmonized to display a margin equal to 1. For our proposed loss, we have averaged the level sets over 100 replications to avoid meshing artifacts.*

Table 4.1: *Summary of standard top-K losses: vanilla top-K ℓ^K; Cross Entropy ℓ_{CE}; hinge top-K ℓ^K_{Hinge}; Convexified hinge top-K $\ell^K_{CVXHinge}$; Calibrated hinge top-K $\ell^K_{Cal.\ Hinge}$; Log-sum Smoothed hinge top-K $\ell^{K,\tau}_{Smoothed\ Hinge}$; Noised balanced hinge top-K $\ell^{K,\epsilon,M}_{Noised\ bal.}$ (**proposed**); Noised imbalanced hinge top-K $\ell^{K,\epsilon,M,m_y}_{Noised\ Imbal.}$ (**proposed**).*

Loss : $\ell(\mathbf{s}, y)$	**Expression**	**Param.**	**Reference**
$\ell^K(\mathbf{s}, y)$	$\mathbb{1}_{\{\tau_K(\mathbf{s}) > s_y\}}$	K	Equation (4.3)
$\ell_{\text{CE}}(\mathbf{s}, y)$	$-\ln\left(e^{s_y} / \sum_{k\in[L]} e^{s_k}\right)$	—	
$\ell^{m_y}_{\text{LDAM}}(\mathbf{s}, y)$	$-\ln\left(e^{s_y - m_y} / \left[e^{s_y - m_y} + \sum_{k\in[L], k\neq y} e^{s_k}\right]\right)$	m_y	[Lin+17]
$\ell^{\gamma}_{\text{focal}}(\mathbf{s}, y)$	$(1 - \log\left[\ell_{\text{CE}}(\mathbf{s}, y)\right])^{\gamma} \ell_{\text{CE}}(\mathbf{s}, y)$	γ	[Cao+19]
$\ell^K_{\text{Hinge}}(\mathbf{s}, y)$	$\left(1 + \tau_K(\mathbf{s}_{\setminus y}) - s_y\right)_+$	K	Equation (4.4), [LHS15]
$\ell^K_{\text{CVXHinge}}(\mathbf{s}, y)$	$\left(\frac{1}{K}\sum_{k\in[K]} \tau_k(\mathbf{1}_L - \delta_y + \mathbf{s}) - s_y\right)_+$	K	[LHS15]
$\ell^K_{\text{Cal. Hinge}}(\mathbf{s}, y)$	$(1 + \tau_{K+1}(\mathbf{s}) - s_y)_+$	K	Equation (4.17), [YK20]
$\ell^{K,\tau}_{\text{Smoothed Hinge}}(\mathbf{s}, y)$	$\tau \ln\left[\sum_{A\subset[L],\lvert A\rvert=K} e^{\frac{\mathbb{1}_{\{y\notin A\}}}{\tau} + \sum_{j\in A}\frac{s_j}{K\tau}}\right] - \tau\ln\left[\sum_{A\subset[L],\lvert A\rvert=K} e^{\sum_{j\in A}\frac{s_j}{K\tau}}\right]$	K, τ	[BZK18]
$\ell^{K,\epsilon,M}_{\text{Noised bal.}}(\mathbf{s}, y)$	$(1 + \widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s}) - s_y)_+$, where $\widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s})$ is the noisy top-$K+1$	K, ϵ, M	Equation (4.8), (**proposed**)
$\ell^{K,\epsilon,M,m_y}_{\text{Noised Imbal.}}(\mathbf{s}, y)$	$(m_y + \widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s}) - s_y)_+$, where $\widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s})$ is the noisy top-$K+1$	K, ϵ, M, m_y	Equation (4.13), (**proposed**)

4.3 Proposed method

4.3.1 Preliminaries

Following classical notation, we deal with multi-class classification that considers the problem of learning a classifier from $\mathcal{X}$ to $[L] \triangleq \{1, \dots, L\}$ based on n pairs of (*input, label*) *i.i.d.* sampled from a joint distribution $\mathbb{P}$: $(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times [L]$, where $\mathcal{X}$ is the input data space ($\mathcal{X}$ is the space of RGB images of a given size in our vision applications) and the y's are the associated labels among L possible ones.

For a training pair of observed features and label (x, y), $\mathbf{s} \in \mathbb{R}^L$ refers to the associated score vector (often referred to as *logits*). From now on, we use bold font to represent vectors. For $k \in [L]$, s_k refers to the score attributed to the k-th class while s_y refers to the score of the true label and $s_{[k]}$ refers to the k-th largest score[1], so that $s_{[1]} \geq \cdots \geq s_{[k]} \geq \cdots \geq s_{[L]}$. For $K \in [L]$, we define τ_K and $\text{top}\Sigma_K$, functions from $\mathbb{R}^L$ to $\mathbb{R}$ as:

$$\tau_K : \mathbf{s} \mapsto s_{[K]} \tag{4.1}$$

$$\text{top}\Sigma_K : \mathbf{s} \mapsto \sum_{k\in[K]} s_{[k]} \ . \tag{4.2}$$

[1]Ties are broken arbitrarily.

In this thesis, τ_K is referred to as the top-K function or top-K operator. We write $\mathbf{1}_L = (1, \dots, 1)^\top \in \mathbb{R}^L$. For $\mathbf{s} \in \mathbb{R}^L$, the gradient $\nabla \tau_K(\mathbf{s})$ is a vector with a single one at the K-th largest coordinate of $\mathbf{s}$ and 0 elsewhere (denoted as $\arg \tau_K(\mathbf{s})$). Similarly, $\nabla \mathrm{top}\Sigma_K(\mathbf{s})$ is a vector with K ones at the K largest coordinates of $\mathbf{s}$ and 0 elsewhere (denoted as $\arg \mathrm{top}\Sigma_K(\mathbf{s})$). The top-$K$ loss (a 0/1 loss) can now be written:

$$\ell^K(\mathbf{s}, y) = \mathbb{1}_{\{\tau_K(\mathbf{s}) > s_y\}} \ . \tag{4.3}$$

This loss reports an error when the score of the true label y is not among the K-th largest scores. One would typically seek to minimize this loss. Yet, being a piece-wise constant function w.r.t. to its first argument[2], numerical difficulties make solving this problem particularly hard in practice. In what follows we recall some popular surrogate top-K losses from the literature before providing new alternatives. We summarize such variants in Table 4.1 and illustrate their differences in Figure 4.1 for $L = 3, K = 2$ (see also Figure 4.6 in Appendix, for $L = 3, K = 1$).

A first alternative introduced by [LHS15] is a relaxation generalizing the multi-class hinge loss introduced by [CS01] to the top-K case:

$$\ell^K_{\text{Hinge}}(\mathbf{s}, y) = \Big(1 + \tau_K(\mathbf{s}_{\setminus y}) - s_y\Big)_+ \ , \tag{4.4}$$

where $\mathbf{s}_{\setminus y}$ is the vector in $\mathbb{R}^{d-1}$ obtained by removing the y-th coordinate of $\mathbf{s}$, and $(\cdot)_+ \triangleq \max(0, \cdot)$. The authors propose a convex loss function ℓ^K_{CVXHinge} (see Table 4.1) which upper bounds the loss function ℓ^K_{Hinge}.

[BZK18] have proposed a smoothed counterpart of ℓ^K_{Hinge}, relying on a recursive algorithm tailored for their combinatorics smoothed formulation. Yet, a theoretical limitation of ℓ^K_{Hinge} and ℓ^K_{CVXHinge} was raised by [YK20] showing that they are not top-K calibrated. Top-K calibration is a property defined by [YK20]. We recall some technical details in Section 4.6.1 and the precise definition of top-K calibration is given in Definition 4.6.2.

We let $\Delta^{L-1} \triangleq \{p \in \mathbb{R}^L : \sum_{l=1}^L p_l = 1, p_l \geq 0\}$ denote the probability simplex of size L. For a score $\mathbf{s} \in \mathbb{R}^L$ and $\mathbf{p} \in \Delta^{L-1}$ representing the conditional distribution of y given x, we write the conditional ℓ-risk at $x \in \mathcal{X}$ as $\mathcal{R}_{\ell|x}(\mathbf{s}, \mathbf{p}) = \mathbb{E}_{y|x\sim p}(\ell(\mathbf{s}, y))$ and the (integrated) ℓ-risk as $\mathcal{R}_\ell(f) \triangleq \mathbb{E}_{(x,y)\sim\mathbb{P}}[\ell(f(x), y)]$ for a scoring function $f : \mathcal{X} \to \mathbb{R}^L$. The associated Bayes risks are defined respectively by $\mathcal{R}^*_{\ell|x}(\mathbf{p}) \triangleq \inf_{\mathbf{s}\in\mathbb{R}^L} \mathcal{R}_{\ell|x}(\mathbf{s}, \mathbf{p})$ and $\mathcal{R}^*_\ell \triangleq \inf_{f:\mathcal{X}\to\mathbb{R}^L} \mathcal{R}_\ell(f)$.

The following result by [YK20] shows that a top-K calibrated loss is top-K consistent, meaning that a minimizer of such a loss would also lead to Bayes optimal classifiers:

Theorem 4.3.1. [*YK20, Theorem 2.2*]**.** *Suppose ℓ is a nonnegative top-K calibrated loss function. Then, ℓ is top-K consistent,* i.e.*, for any sequence of measurable functions $f^{(n)} : \mathcal{X} \to \mathbb{R}^L$, we have:*

$$\mathcal{R}_\ell\left(f^{(n)}\right) \to \mathcal{R}^*_\ell \implies \mathcal{R}_{\ell^K}\left(f^{(n)}\right) \to \mathcal{R}^*_{\ell^K} \ .$$

[2]See for instance Figure 4.1(a) for a visualization.

In their paper, [YK20] propose a slight modification of the multi-class hinge loss ℓ^K_{Hinge} and show that it is top-K calibrated:

$$\ell^K_{\text{Cal. Hinge}}(\mathbf{s}, y) = (1 + \tau_{K+1}(\mathbf{s}) - s_y)_+ \ . \tag{4.5}$$

The loss $\ell^K_{\text{Cal. Hinge}}$ thus has an appealing theoretical guarantee that ℓ^K_{Hinge} does not have. Therefore we will use $\ell^K_{\text{Cal. Hinge}}$ as the starting point of our smoothing proposal.

4.3.2 New loss for balanced top-K classification

[BZK18] have shown experimentally that a deep learning model trained with ℓ^K_{Hinge} does not learn. The authors claim that the reason for this is the non smoothness of the loss and the sparsity of its gradient.

We also show in Table 4.2 that a deep learning model trained with $\ell^K_{\text{Cal. Hinge}}$ yields poor results. The problematic part stems from the top-K function which is non-smooth and whose gradient has only one non-zero element (that is equal to one). In this chapter we propose to smooth the top-K function with the *perturbed optimizers* method developed by [Ber+20]. We follow this strategy due to its flexibility and to the ease of evaluating associated first order information (a crucial point for deep neural network frameworks).

Definition 4.3.2. For a smoothing parameter $\epsilon > 0$, we define for any $\mathbf{s} \in \mathbb{R}^L$ the ϵ-smoothed version of $\text{top}\Sigma_K$ as:

$$\text{top}\Sigma_{K,\epsilon}(\mathbf{s}) \triangleq \mathbb{E}_Z[\text{top}\Sigma_K(\mathbf{s} + \epsilon Z)] \ , \tag{4.6}$$

where Z is a standard normal random vector, *i.e.*, $Z \sim \mathcal{N}(0, \text{Id}_L)$.

Proposition 4.3.3. *For a smoothing parameter $\epsilon > 0$,*

- *The function $\text{top}\Sigma_{K,\epsilon} : \mathbb{R}^L \to \mathbb{R}$ is strictly convex, twice differentiable and $\sqrt{K}$-Lipschitz continuous.*
- *The gradient of $\text{top}\Sigma_{K,\epsilon}$ reads:*

$$\nabla_{\mathbf{s}}\text{top}\Sigma_{K,\epsilon}(\mathbf{s}) = \mathbb{E}[\arg\text{top}\Sigma_K(\mathbf{s} + \epsilon Z)] \ . \tag{4.7}$$

- *$\nabla_{\mathbf{s}}\text{top}\Sigma_{K,\epsilon}$ is $\frac{\sqrt{KL}}{\epsilon}$-Lipschitz.*
- *When $\epsilon \to 0$, $\text{top}\Sigma_{K,\epsilon}(\mathbf{s}) \to \text{top}\Sigma_K(\mathbf{s})$.*

All proofs are given in the appendix.

The smoothing strategy introduced leads to a natural smoothed approximation of the top-K operator, leveraging the link $\tau_K(\mathbf{s}) = \text{top}\Sigma_K(\mathbf{s}) - \text{top}\Sigma_{K-1}(\mathbf{s})$ for any score $\mathbf{s} \in \mathbb{R}^L$ (where we use the convention $\text{top}\Sigma_0 = \mathbf{0}_L \in \mathbb{R}^L$):

Definition 4.3.4. For any $s \in \mathbb{R}^L$ and $k \in [L]$, we define

$$\tau_{K,\epsilon}(\mathbf{s}) \triangleq \text{top}\Sigma_{K,\epsilon}(\mathbf{s}) - \text{top}\Sigma_{K-1,\epsilon}(\mathbf{s}) \ .$$

This definition leads to a smooth approximation of the top-K function, in the following sense:

Proposition 4.3.5. *For a smoothing parameter $\epsilon > 0$,*

- *$\tau_{K,\epsilon}$ is $\frac{4\sqrt{KL}}{\epsilon}$-smooth.*
- *For any $\mathbf{s} \in \mathbb{R}^L$, $|\tau_{K,\epsilon}(\mathbf{s}) - \tau_K(\mathbf{s})| \leq \epsilon \cdot C_{K,L}$, where $C_{K,L} = K\sqrt{2 \log L}$.*

Observe that the last point implies that for any $\mathbf{s} \in \mathbb{R}^L$, $\tau_{K,\epsilon}(\mathbf{s}) \to \tau_K(\mathbf{s})$ when $\epsilon \to 0$.

We can now define an approximation of the calibrated top-K hinge loss $\ell^K_{\text{Cal. Hinge}}$ using $\tau_{K,\epsilon}$ in place of τ_K (see Figures 4.1(h) and 4.1(i) for level sets with $K = 2$)[3].

Definition 4.3.6. We define $\ell^{K,\epsilon}_{\text{Noised bal.}}$ the noised balanced top-K hinge loss as:

$$\ell^{K,\epsilon}_{\text{Noised bal.}}(\mathbf{s}, y) = (1 + \tau_{K+1,\epsilon}(\mathbf{s}) - s_y)_+ \ . \tag{4.8}$$

We call the former balanced as the margin (equal to 1) is the same for all L classes. The parameter ϵ controls the variance of the noise added to the score vectors. When $\epsilon = 0$, we recover the top-K calibrated loss of [YK20], $\ell^K_{\text{Cal. Hinge}}$.

Proposition 4.3.7. *For a smoothing parameter $\epsilon > 0$ and a label $y \in [L]$, • $\ell^{K,\epsilon}_{\text{Noised bal.}}(\cdot, y)$ is continuous, differentiable almost everywhere, with continuous derivative. • The gradient of $\ell(\cdot, y) \triangleq \ell^{K,\epsilon}_{\text{Noised bal.}}(\cdot, y)$ is given by:*

$$\nabla \ell(\mathbf{s}, y) = \mathbb{1}_{\{1+\tau_{K+1,\epsilon}(\mathbf{s}) \geq s_y\}} \cdot (\nabla \tau_{K+1,\epsilon}(\mathbf{s}) - \delta_y), \tag{4.9}$$

where $\delta_y \in \mathbb{R}^L$ is the vector with 1 at coordinate y and 0 elsewhere.

Practical implementation: As is, the proposed loss can not be used directly to train modern neural network architectures due to the expectation and remains a theoretical tool. Following [Ber+20], we simply rely on a Monte Carlo method to estimate the expectation for both the loss and its gradient: we draw M noise vectors $Z_1, \ldots, Z_M$, with $Z_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \mathrm{Id}_L)$ for $m \in [M]$. The loss $\ell^{K,\epsilon}_{\text{Noised bal.}}$ is then estimated by:

$$\ell^{K,\epsilon,M}_{\text{Noised bal.}}(\mathbf{s}, y) = (1 + \widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s}) - s_y)_+ \ , \tag{4.10}$$

where $\widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s}) \triangleq \widehat{\text{top}\Sigma}_{K+1,\epsilon,M}(\mathbf{s}) - \widehat{\text{top}\Sigma}_{K,\epsilon,M}(\mathbf{s})$ is a Monte Carlo estimate with M samples:

$$\widehat{\text{top}\Sigma}_{K,\epsilon,M}(\mathbf{s}) = \frac{1}{M} \sum_{m=1}^{M} \text{top}\Sigma_K(\mathbf{s} + \epsilon Z_m) \ . \tag{4.11}$$

We approximate $\nabla_{\mathbf{s}} \ell^{K,\epsilon}_{\text{Noised bal.}}(\mathbf{s}, y)$ by G, with:

$$G = \mathbb{1}_{\{1+\widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s}) \geq s_y\}} \cdot (\widehat{\nabla \tau}_{K+1,\epsilon,M}(\mathbf{s}) - \delta_y), \tag{4.12}$$

where the Monte Carlo estimate

$$\widehat{\nabla \tau}_{K+1,\epsilon,M}(\mathbf{s}) \triangleq \widehat{\arg \tau}_{K+1,\epsilon,M}(\mathbf{s})$$

[3]For illustrations with $K = 1$, see Figures 4.6(h) and 4.6(j)

is given by:

$$\widehat{\arg\tau}_{K+1,\epsilon,M}(\mathbf{s}) = \frac{1}{M}\sum_{m=1}^{M} \arg\tau_{K+1}(\mathbf{s}+\epsilon Z_m) \ .$$

We train our loss with SGD. Hence, M repetitions are drawn each time the loss is evaluated. The iterative nature of this process helps amplify the smoothing power of the approach, explaining why even small values of M can lead to good performance (see Section 5.6).

Illustration. Consider the case $L = 4$, $K = 2$, $M = 3$, $\epsilon = 1.0$ with a score vector $\mathbf{s} = \begin{bmatrix} 2.4 \\ 2.6 \\ 2.3 \\ 0.5 \end{bmatrix}$. We have $\tau_K(\mathbf{s}) = 2.4$ and $\arg\tau_K(\mathbf{s}) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ (the second largest component of s corresponds to the first coordinate). Assume the three noise vectors sampled are:

$$Z_1 = \begin{bmatrix} 0.2 \\ -0.1 \\ 0.1 \\ 0.3 \end{bmatrix}, \ Z_2 = \begin{bmatrix} 0.1 \\ 0.1 \\ -0.1 \\ 0.1 \end{bmatrix}, \ Z_3 = \begin{bmatrix} -0.1 \\ -0.1 \\ 0.1 \\ -0.1 \end{bmatrix}.$$

The perturbed vectors are now:

$$\mathbf{s}+\epsilon Z_1 = \begin{bmatrix} 2.6 \\ 2.5 \\ 2.4 \\ 0.8 \end{bmatrix}, \ \mathbf{s}+\epsilon Z_2 = \begin{bmatrix} 2.5 \\ 2.7 \\ 2.2 \\ 0.6 \end{bmatrix}, \ \mathbf{s}+\epsilon Z_3 = \begin{bmatrix} 2.3 \\ 2.5 \\ 2.4 \\ 0.4 \end{bmatrix}.$$

The induced perturbation may provoke a change in both τ_K and $\arg\tau_K$. For the perturbed vector $\mathbf{s}+\epsilon Z_2$, the added noise changes the second largest value but it is still achieved at coordinate 1: $\tau_K(\mathbf{s}+\epsilon Z_2) = 2.5$ and $\arg\tau_K(\mathbf{s}+\epsilon Z_2) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. However, for $\mathbf{s}+\epsilon Z_1$ and $\mathbf{s}+\epsilon Z_3$, the added noise changes the coordinate of the second largest value: $\arg\tau_K(\mathbf{s}+\epsilon Z_1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and $\arg\tau_K(\mathbf{s}+\epsilon Z_3) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, with $\tau_K(\mathbf{s}+\epsilon Z_1) = 2.5$ and $\tau_K(\mathbf{s}+\epsilon Z_3) = 2.4$, giving:

$$\widehat{\tau}_{K,\epsilon,M}(s) = (2.5 + 2.5 + 2.4)/3 = 2.47 \ ,$$

$$\widehat{\nabla\tau}_{K,\epsilon,M}(s) = \frac{1}{3}\left(\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \\ 0 \end{bmatrix}.$$

We see the added noise results in giving weight to the gradient coordinates k whose associated score s_k is close to $\tau_K(\mathbf{s})$ (in this example the first and third coordinates). Note that if we set ϵ to a smaller value, *e.g.*, $\epsilon = 0.1$, the added perturbation is not large enough to change the $\arg\tau_K$ in the perturbed vectors, leading to the same gradient as the non-smoothed top-K operator: $\widehat{\nabla\tau}_{K,0.1}(s) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. Hence, ϵ acts as a parameter which allows exploring coordinates k whose score values s_k are close to the top-K score (provided that ϵ and/or M are large enough).

4.3.3 New loss for imbalanced top-K classification

In real world applications, a long-tailed distribution between the classes is often present, *i.e.*, few classes receive most of the annotated labels. This occurs for instance in datasets

Table 4.2: *Influence of ϵ on CIFAR-100 best validation top-5 accuracy obtained by training a DenseNet 40-40 with loss $\ell^{K=5,\epsilon,M=10}_{\text{Noised bal.}}$. The training procedure is the same as in Section 4.4.4.*

ϵ	0.0	1e-4	1e-3	1e-2	1e-1	1.0	10.0	100.0
Top-5 acc.	19.38	14.84	11.4	93.36	94.46	94.24	93.78	93.12

such as Pl@ntNet-300K [Gar+21] and Inaturalist [Hor+18], where a few classes represent the vast majority of images. For these cases, the performance of deep neural networks trained with the cross entropy loss is much lower for classes with a small numbers of images, see [Gar+21].

We present an extension of the loss presented in Section 4.3.2 to the imbalanced case. This imbalanced loss is based on uneven margins [Sco12; LST03; IMV19; Cao+19]. The underlying idea is to require larger margins for classes with few examples, which leads to a higher incurred loss for mistakes made on examples of the least common classes.

Imposing a margin m_y parameter per class in Equation (4.10) leads to the following formulation:

$$\ell^{K,\epsilon,M,m_y}_{\text{Noised Imbal.}}(\mathbf{s}, y) = (m_y + \widehat{\tau}_{K+1,\epsilon,M}(\mathbf{s}) - s_y)_+ \quad . \tag{4.13}$$

Here, we follow [Cao+19] and set $m_y = C/n_y^{1/4}$, with n_y the number of samples in the training set with class y, and C a hyperparameter to be tuned on a validation set.

4.3.4 Comparisons of various top-K losses

In Table 4.1, we synthesize the various top-K loss functions evoked above. To better understand their differences, Figure 4.1 provides a plot of the losses for $\mathbf{s}$ in the 2-simplex, for $K = 2$ and $L = 3$. The correct label is set to be $y = 3$ and corresponds to the vertex on top and in red. Figure 4.1(a) shows the classical top-K that we would ideally want to optimize. It has 0 error when s_3 is larger than the smallest coordinate of $\mathbf{s}$ (*i.e.*, is in the top-2) and 1 otherwise. Figure 4.1(b) shows the cross-entropy, by far the most popular (convex) loss used in deep learning. As mentioned by [YK20], the cross-entropy happens to be top-K calibrated for all K. Figure 4.1(c) shows the top-K hinge loss proposed by [LHS15] and Figure 4.1(e) is a convex upper relaxation. Unfortunately, [YK20] have shown that such losses are not top-K calibrated and propose an alternative, illustrated in Figure 4.1(d). We show that the loss of [YK20] performs poorly when used for optimizing a deep neural network. Figure 4.1(f) and Figure 4.1(g) show the smoothing proposed by [BZK18] of the loss in Figure 4.1(c), while Figure 4.1(h) and Figure 4.1(i) show our proposed noised smoothing of the loss in Figure 4.1(d). The difference with [BZK18] is that we start with a top-K calibrated hinge loss, and our smoothing consists in smoothing only the top-K operator, which mostly affects classes whose scores are close to the top-K score, while the method from [BZK18] results in a gradient where all coordinates are non-zero.

Finally, Figure 4.1(j) shows our noised imbalanced top-K loss. Additional visualizations of our noised top-K loss illustrating the effect of M and ϵ can be found in the appendix, see Figures 4.4 and 4.5.

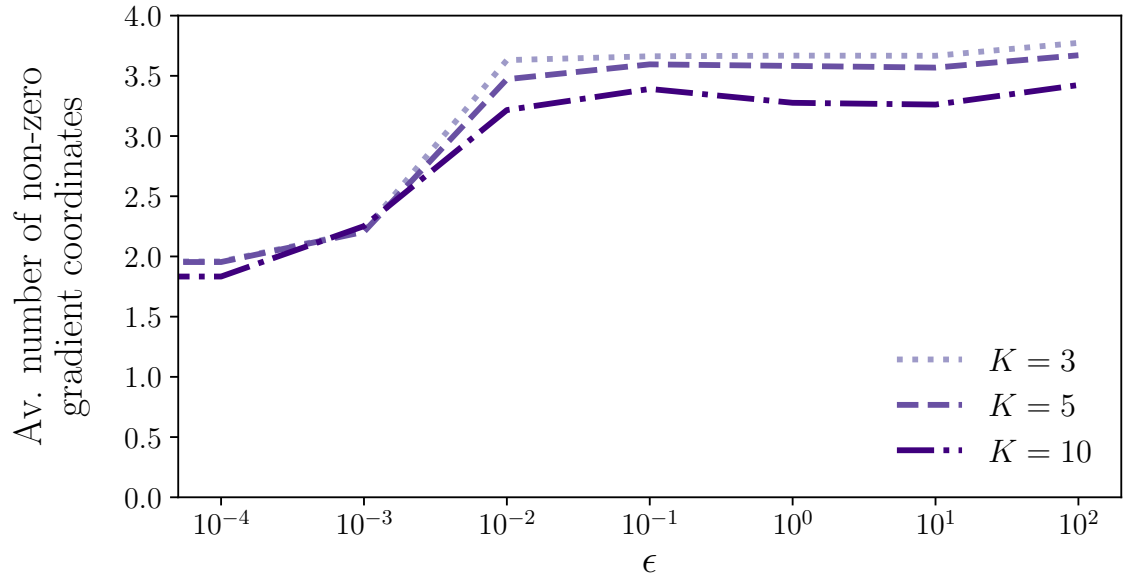


Figure 4.2: *Average number of non-zero gradient coordinates as a function of ϵ (loss $\ell^{K,\epsilon,3}_{\text{Noised bal.}}$, CIFAR-100 dataset, DenseNet 40-40 model, 1st epoch). The gradient dimension is 100. We see that the gradient remains sparse even for large values of ϵ. Together with Table 4.2, this shows that having a non-sparse gradient is not a necessary condition for successful learning, contrary to what is suggested in [BZK18]*

4.4 Experiments

The Pytorch [Pas+19] code for our top-K loss and experiments can be found at: `https://github.com/garcinc/noised-topk`. We also provide a python package containing our top-K losses as well as the smoothed top-K function. The package is easily installed with pip:

```
pip install pytopk
```

4.4.1 Influence of ϵ and gradient sparsity

Table 4.2 shows the influence of ϵ on CIFAR-100[4] top-5 accuracy. When $\epsilon = 0$, $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ coincides with $\ell^{K}_{\text{Cal. Hinge}}$. We see that a model trained with this loss fails to learn properly. This resonates with the observation made by [BZK18] that a model trained with ℓ^{K}_{Hinge}, which is close to $\ell^{K}_{\text{Cal. Hinge}}$, also fails to learn. Table 4.2 also shows that when ϵ is too small ($\epsilon = 10^{-4}$ or $\epsilon = 10^{-3}$), the optimization remains difficult and the learned models have very low performance. For sufficiently high values of ϵ, in the order of 10^{-2} or greater, the smoothing is effective and the learned models achieve a very high top-5 accuracy. Table 4.2 also shows that although the optimal value of ϵ appears to be around 10^{-1}, the optmization is robust to high values of ϵ.

[BZK18] argue that a reason a model trained with ℓ^{K}_{Hinge} fails to learn is because of the sparsity of the gradient of ℓ^{K}_{Hinge}. Indeed, for any $\mathbf{s} \in \mathbb{R}^L$, $y \in [L]$, $\nabla_s \ell^{K}_{\text{Hinge}}(\mathbf{s}, y)$ has at most two non-zero coordinates. This is one of the main reasons put forward

[4]For experiments with CIFAR-100, we consider a DenseNet 40-40 model [Hua+17], similarly as [BZK18].

Table 4.3: *Influence of M hyper-parameter on the best validation top-5 accuracy (loss $\ell^{5,0.2,M}_{\text{Noised bal.}}$, CIFAR-100 dataset, DenseNet 40-40 model. The training procedure is the same as in Section 4.4.4.)*

M	1	2	3	5	10	50	100
Top-5 acc	94.28	94.2	94.46	94.52	94.24	94.64	94.52

by the authors to motivate the smoothing of ℓ^K_{Hinge} into $\ell^{K,\tau}_{\text{Smoothed Hinge}}$, whose gradient coordinates are all non-zero.

We investigate the behaviour of the gradient of our loss by computing $\nabla_s \ell^{K,\epsilon,M}_{\text{Noised bal.}}(\mathbf{s}, y)$ for each training example during the first epoch. We then compute the average number of non-zero coordinates in the gradient. We repeat this process for several values of ϵ and report the results in Figure 4.2. There are two points to highlight:

• The number of non-zero gradients coordinates increases with ϵ. This is consistent with our illustration example in Section 4.3.2: high values of ϵ allow putting *weights* on gradient coordinates whose score is close to the K-th largest score.

• Even when ϵ is large, the number of non-zero gradient coordinates is small: on average, 4 out of 100. In comparison, for ℓ^K_{CE} and $\ell^{K,\tau}_{\text{Smoothed Hinge}}$, all gradient coordinates are non-zero. Yet, even with such sparse gradient vectors, we manage to reach better top-5 accuracies than ℓ^K_{CE} and $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ (see Table 4.4). Therefore, one of the main takeaway is that a non-sparse gradient does not appear to be a necessary condition for successful learning contrary to what is suggested in [BZK18]. A sufficiently high probability (controlled by ϵ) that each coordinate is updated at training is enough to achieve good performance.

4.4.2 Influence of M

Table 4.3 shows the influence of the number of sampled standard normal random vectors M on CIFAR-100 top-5 accuracy for a model trained with our balanced loss with $K = 5$. M appears to have little impact on top-5 accuracy, indicating that there is no need to precisely estimate the expectation in Equation (4.11). As increasing M comes with computation overhead (see next section) and does not yield an increase of top-K accuracy, we advise setting it to a small value (*e.g.*, $3 \leq M \leq 10$).

4.4.3 Computation time

In Figure 4.3, we plot the average epoch duration of a model trained with the cross entropy ℓ_{CE}, the loss from [BZK18] $\ell^{K,\tau}_{\text{Smoothed Hinge}}$, and our balanced loss $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ (for several values of M) as a function of K. For standard training values, *i.e.*, $K = 5$, $M = 3$, the average epoch time is 65s for ℓ_{CE}, 68s for $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ (+4.6% w.r.t. ℓ_{CE}) and 81s for $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ (+24.6% w.r.t. ℓ_{CE}). Figure 4.3 further shows that while the average epoch duration of $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ seems to scale linearly in K, our loss does not incur an increased epoch duration when K increases. Thus, for $K = 10$, the average epoch time

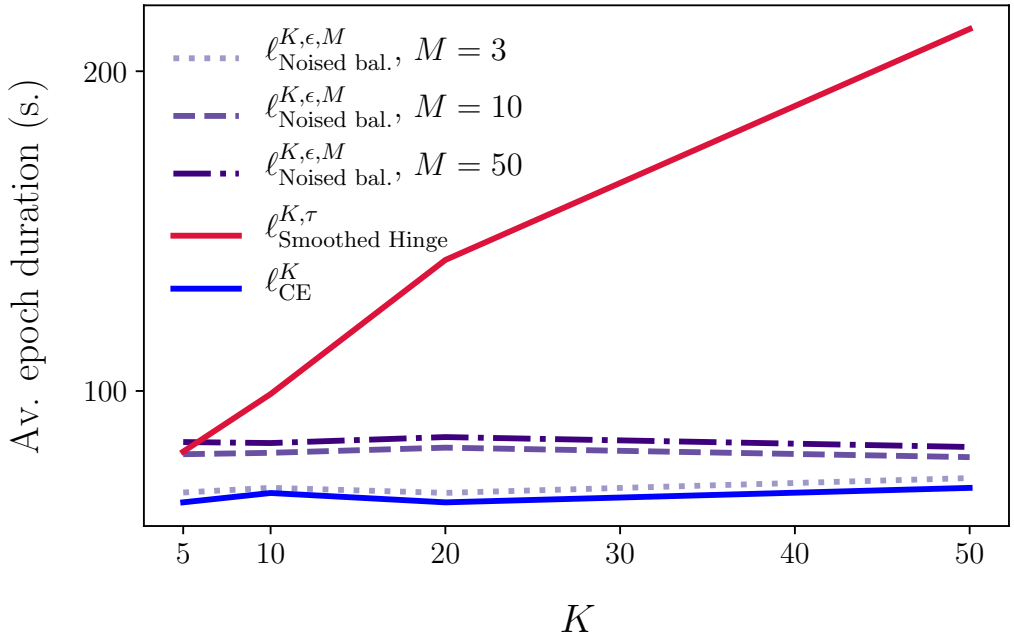


Figure 4.3: *Average epoch time as a function of K for different losses (CIFAR-100 dataset, DenseNet 40-40 model). The proposed loss $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ is not sensitive to the parameter K contrarily to $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ introduced by [BZK18].*

for $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ is 90s versus 60s for $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ with $M = 3$. While for most classical datasets [Rus+15; Kri09] small values of K are enough to achieve high top-K accuracy, for other applications high values of K may be used [CAS16; Col+20], making our balanced loss computationally attractive.

4.4.4 Comparisons for balanced classification

The CIFAR-100 [Kri09] dataset contains 60,000 images (50,000 images in the training set and 10,000 images in the test set) categorized in 100 classes. The classes are grouped into 20 superclasses (*e.g.*, fish, flowers, people), each regrouping 5 classes. Here we compare the top-K accuracy of a deep learning model trained on CIFAR-100 with either our balanced loss $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$, the cross entropy ℓ_{CE}, or the loss from [BZK18], $\ell^{K,\tau}_{\text{Smoothed Hinge}}$, which is the current state-of-the art top-K loss for deep learning. We repeat the experiment from Section 5.1 of [BZK18] to study how $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ reacts when noise is introduced in the labels. More precisely, for each training image, its label is sampled randomly within the same super-class with probability p. Thus, $p = 0$ corresponds to the original dataset and $p = 0.5$ corresponds to a dataset where all the training examples have a label corresponding to the right superclass, but half of them (on average) have a different label than the original one. With such a dataset, a perfect top-5 classifier is expected to have a 100% top-5 accuracy. As in [BZK18] we extract 5000 images from the training set to build a validation set. We use the same

Table 4.4: *Top-5 accuracy for different losses as a function of the label noise probability p within the superclasses of CIFAR-100 (DenseNet 40-40 model).*

Label noise p	ℓ^K_{CE}	$\ell^{5,1.0}_{\text{Smoothed Hinge}}$	$\ell^{5,0.2,10}_{\text{Noised bal.}}$
0.0	94.2±0.1	93.3±0.0	**94.4**±0.1
0.1	90.3±0.2	**92.2**±0.3	91.9±0.1
0.2	87.6±0.1	90.4±0.2	**90.7**±0.5
0.3	85.7±0.4	88.8±0.1	**89.7**±0.1
0.4	83.6±0.2	87.4±0.1	**87.8**±0.6

Table 4.5: *Macro-average top-K accuracy (on test set) for different losses measured on Pl@ntNet-300K, a heavy-tailed dataset with high ambiguity (ResNet-50 model). The three numbers in parentheses represent respectively the mean top-K accuracies of 1) few shot classes (< 20 training images) 2) medium shot classes ($20 \leq . \leq 100$ training images) 3) many shot classes (> 100 training images).*

K	1	3	5
ℓ_{CE}	36.3±0.3 (12.6/42.9/71.7)	58.8±0.4 (32.4/**75.3**/92.0)	68.7±0.2 (45.1/**86.3**/95.4)
$\ell^{K,0.1}_{\text{Smoothed Hinge}}$	35.7±0.2 (13.1/41.5/71.1)	50.3±0.2 (16.7/69.8/**92.7**)	50.9±0.3 (12.1/78.1/95.7)
$\ell^{K,1.0,5}_{\text{Noised bal.}}$	35.8±0.3 (12.4/42.1/**72.1**)	58.7±0.4 (32.2/73.8/88.8)	66.4±0.5 (42.0/82.5/95.5)
focal ($\gamma = 2.0$)	37.6±0.3 (15.5/43.4/71.4)	60.4±0.3 (35.9/74.8/92.0)	69.7±0.2 (47.5/84.8/**95.8**)
$\ell^{\max m_y=0.2}_{\text{LDAM}}$	40.6±0.1 (20.9/45.8/71.2)	63.3±0.3 (43.0/74.1/90.0)	71.9±0.3 (54.0/83.0/94.0)
$\ell^{K,0.01,5,\max m_y=0.2}_{\text{Noised imbal.}}$	**42.4**±0.3 (**23.9**/**46.3**/72.1)	**64.9**±0.4 (**44.8**/74.5/92.1)	**73.2**±0.5 (**55.3**/84.2/95.3)

hyperparameters as [BZK18]: we train a DenseNet 40-40 [Hua+17] for 300 epochs with SGD and a Nesterov momentum of 0.9. For the learning rate, our policy consists in starting with value of 0.1 and dividing it by ten at epoch 150 and 225. The batch size and weight decay are set to 64 and 1.10^{-4} respectively. Following [BZK18], the smoothing parameter τ of $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ is set to 1.0. For $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$, we set the noise parameter ϵ to 0.2 and the number of noise samples M at 10. We keep the models with the best top-5 accuracies on the validation set and report the top-5 accuracy on the test set in Table 4.4. The results are averaged over four runs with four different random seeds (we give the 95% confidence inverval). They show that the models trained with $\ell^{5,0.2,10}_{\text{Noised bal.}}$ give the best top-5 accuracies except when $p = 0.1$ where it is slightly below $\ell^{5,1.0}_{\text{Smoothed Hinge}}$. We also observe that the performance gain over the cross entropy is significant in the presence of label noise (*i.e.,* for $p > 0$).

We provide additional experiments on ImageNet [Rus+15] in Section 4.6.4.

4.4.5 Comparison for imbalanced classification

4.4.5.1 Pl@ntNet-300K

We consider Pl@ntNet-300K[5], a dataset of plant images recently introduced in [Gar+21]. It consists of 306,146 plant images distributed in 1,081 species (the classes). The

[5]For the experiments with Pl@ntNet-300K, we consider a ResNet-50 model [He+16].

Table 4.6: *Top-K accuracy (on test set) for different losses measured on ImageNet-LT (ResNet-34 model). The three numbers in parentheses represent respectively the mean top-K accuracies of 1) few shot classes (< 20 training images) 2) medium shot classes ($20 \leq . \leq 100$ training images) 3) many shot classes (> 100 training images).*

K	1	3	5
ℓ_{CE}	37.0±0.1 (1.5/28.2/60.6)	55.5±0.1 (8.2/53.0/**75.3**)	63.2±0.1 (15.8/63.1/80.1)
$\ell^{K,0.1}_{\text{Smoothed Hinge}}$	37.3±0.1 (1.3/28.6/**60.7**)	42.0±0.1 (0.0/29.1/72.9)	39.0±0.1 (0.0/20.7/75.5)
focal ($\gamma = 1.0$)	37.7±0.0 (2.4/29.8/59.9)	56.2±0.0 (10.2/**54.0**/75.1)	**63.8**±0.1 (17.8/**63.6**/**80.3**)
$\ell^{\max m_y=0.4}_{\text{LDAM}}$	**39.3**±0.2 (**10.5**/**33.1**/57.1)	56.0±0.2 (24.1/52.0/72.3)	63.1±0.2 (32.9/60.1/77.7)
$\ell^{K,0.1,5,\max m_y=0.4}_{\text{Noised imbal.}}$	38.7±0.0 (7.6/32.3/57.6)	**56.5**±0.1 (**27.0**/52.6/71.7)	63.5±0.1 (**37.0**/60.2/77.0)

particularities of the dataset are its long-tailed distribution (80% of the species with the least number of images account for only 11% of the total number of images) and the class ambiguity: many species are visually similar. For such an imbalanced dataset, accuracy and top-K accuracy mainly reflect the performance of the model on the few classes representing the vast majority of images. Often times, we also want the model to yield satisfactory results on the classes with few images. Therefore, for this dataset we report macro-average top-K accuracy, which is obtained by computing top-K accuracy for each class separately and then taking the average over classes. Thus, the class with only a few images contributes the same as the class with thousands of images to the overall result.

In this section we compare the macro-average top-K accuracy of a deep neural network trained with either ℓ_{CE}, $\ell^{K,\tau}_{\text{Smoothed Hinge}}$, $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$, $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$ and $\ell^{m_y}_{\text{LDAM}}$, the loss from [Cao+19] based on uneven margins providing state-of-the-art performance in Fine-Grained Visual Categorization tasks.

Setup: We train a ResNet-50 [He+16] pre-trained on ImageNet [Rus+15] for 30 epochs with SGD with a momentum of 0.9 with the Nesterov acceleration. We use a learning rate of 2.10^{-3} divided by ten at epoch 20 and epoch 25. The batch size and weight decay are set to 32 and 1.10^{-4} respectively. The smoothing parameter τ for $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ is set to 0.1.

To tune the margins of $\ell^{m_y}_{\text{LDAM}}$ and $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$ more easily, we follow [Wan+18; Cao+19]: we normalize the last hidden activation and the weight vectors of the last fully-connected layer to both have unit ℓ_2-norm, and we multiply the scores by a scaling constant, tuned for both losses on the validation set, leading to 40 for $\ell^{m_y}_{\text{LDAM}}$ and 60 for $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$.

Finally, we tune the constant C by tuning the largest margin $\max_{y\in[L]} m_y$ for both $\ell^{m_y}_{\text{LDAM}}$ and $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$. We find that for both losses, the optimal largest margin is 0.2. For $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$, we set $\epsilon = 10^{-2}$ and $M = 5$. We further discuss hyperparameter tuning in Section 4.6.5.

We train the network with the top-K losses $\ell^{K,\tau}_{\text{Smoothed Hinge}}$, $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ and $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$ for $K \in \{1, 3, 5\}$. For all losses, we perform early stopping based on the best macro-average top-K accuracy on the validation set (for $K \in \{1, 3, 5\}$). We report the results on the

test set in Table 4.5 (three seeds, 95% confidence interval). We find that the loss from [BZK18] fails to generalize to the tail classes in such an imbalanced setting. In contrast, $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ gives results similar to the cross-entropy while $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$ provides the best results (regardless of the value of K). Noticeably, it outperforms $\ell^{m_y}_{\text{LDAM}}$ [Cao+19] for all cases.

4.4.5.2 ImageNet-LT

We test $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$ on ImageNet-LT [Liu+19], a dataset of 115,836 training images obtained by subsampling images from ImageNet with a pareto distribution. The resulting class imbalance is much less pronounced than for Pl@ntNet-300K.

We train a ResNet34 with several losses for $K \in \{1, 3, 5\}$ for 100 epochs with a learning rate of 1.10^{-2} divided by 10 at epoch 60 and 80. We use a batch size of 128 and set the weight decay to 2.10^{-3}. All hyperparameters are tuned on the 20,000 images validation set from [Liu+19]. The results on the test set (four seeds, 95% confidence interval) are reported in Table 4.6. They show that $\ell^{K,\epsilon,M,m_y}_{\text{Noised Imbal.}}$ performs very well on few shot classes compared to the other losses. Since ImageNet-LT is much less imbalanced than Pl@ntNet-300K, there are fewer such classes, hence the overall gain is less salient than for Pl@ntNet-300K.

4.5 Conclusion and perspectives

We propose a novel top-K loss as a smoothed version of the top-K calibrated hinge loss of [YK20]. Our loss function is well suited for training deep neural networks, contrarily to the original top-K calibrated hinge loss (*e.g.*, the poor performance of the case $\epsilon = 0$ in Table 4.2, that reduces to their loss). The smoothing procedure we propose applies the perturbed optimizers framework to smooth the top-K operator. We show that our loss performs well compared to the current state-of-the-art top-K losses for deep learning while being significantly faster to train when K increases. At training, the gradient of our loss w.r.t. the score is sparse, showing that non-sparse gradients are not necessary for successful learning. Finally, a slight adaptation of our loss for imbalanced datasets (leveraging uneven margins) outperforms other baseline losses. Studying deep learning optimization methods for other set-valued classification tasks, such as *average size control* or *point-wise error control* [Chz+21] are left for future work.

4.6 Appendix

4.6.1 Reminder on Top-K calibration

Here we provide some elements introduced by [YK20] on top-K calibration.

Definition 4.6.1. [YK20, Definition 2.3]**.** For a fixed $K \in [L]$, and given $\mathbf{y} \in \mathbb{R}^L$ and $\tilde{\mathbf{y}} \in \mathbb{R}^L$, we say that $\mathbf{y}$ is top-K preserving w.r.t. $\tilde{\mathbf{y}}$, denoted $P_K(\mathbf{y}, \tilde{\mathbf{y}})$, if for all $k \in [L]$,

$$\tilde{y}_k > \tau_{K+1}(\tilde{\mathbf{y}}) \implies y_k > \tau_{K+1}(\mathbf{y}) \tag{4.14}$$
$$\tilde{y}_k < \tau_K(\tilde{\mathbf{y}}) \implies y_k < \tau_K(\mathbf{y}) \ . \tag{4.15}$$

The negation of this statement is $\neg P_k(\mathbf{y}, \tilde{\mathbf{y}})$.

We let $\Delta^{L-1} \triangleq \{\mathbf{p} \in \mathbb{R}^L : \sum_{k\in[L]} p_k = 1, p_k \geq 0\}$ denote the probability simplex of size L. For a score $\mathbf{s} \in \mathbb{R}^L$ and $\mathbf{p} \in \Delta^{L-1}$ representing the conditional distribution of y given x, we write the conditional risk at $x \in \mathcal{X}$ as $\mathcal{R}_{\ell|x}(\mathbf{s}, \mathbf{p}) = \mathbb{E}_{y|x\sim p}(\ell(\mathbf{s}, y))$ and the (integrated) risk as $\mathcal{R}_\ell(f) \triangleq \mathbb{E}_{(x,y)\sim\mathbb{P}}[\ell(f(x), y)]$ for a scoring function $f : \mathcal{X} \to \mathbb{R}^L$. The associated Bayes risks are defined respectively by $\mathcal{R}^*_{\ell|x}(\mathbf{p}) \triangleq \inf_{\mathbf{s}\in\mathbb{R}^L} \mathcal{R}_{\ell|x}(\mathbf{s}, \mathbf{p})$ and $\mathcal{R}^*_\ell \triangleq \inf_{f:\mathcal{X}\to\mathbb{R}^L} \mathcal{R}_\ell(f)$.

Definition 4.6.2. [YK20, Definition 2.4]**.** A loss function $\ell : \mathbb{R}^L \times \mathcal{Y} \to \mathbb{R}$ is top-K calibrated if for all $\mathbf{p} \in \Delta^{L-1}$ and all $x \in \mathcal{X}$:

$$\inf_{\mathbf{s}\in\mathbb{R}^L : \neg P_k(\mathbf{s},\mathbf{p})} \mathcal{R}_{\ell|x}(\mathbf{s}, \mathbf{p}) > \mathcal{R}^*_{\ell|x}(\mathbf{p}) \ . \tag{4.16}$$

In other words, a loss is calibrated if the infimum can only be attained among top-K preserving vectors w.r.t. the conditional probability distribution.

Theorem 4.6.3. *Suppose ℓ is a nonnegative top-K calibrated loss function. Then ℓ is top-K consistent*, i.e., *for any sequence of measurable functions $f^{(n)} : \mathcal{X} \to \mathbb{R}^L$, we have:*

$$\mathcal{R}_\ell\left(f^{(n)}\right) \to \mathcal{R}^*_\ell \implies \mathcal{R}_{\ell^K}\left(f^{(n)}\right) \to \mathcal{R}^*_{\ell^K} \ .$$

In their paper, [YK20] propose a slight modification of the multi-class hinge loss ℓ^K_{Hinge} and show that it is top-K calibrated:

$$\ell^K_{\text{Cal. Hinge}}(\mathbf{s}, y) = (1 + \tau_{K+1}(\mathbf{s}) - s_y)_+ \ . \tag{4.17}$$

4.6.2 Proofs and technical lemmas

4.6.2.1 Proof of Proposition 4.3.3

Proof. We define $\Delta^{L-1}_K \triangleq \left\{\mathbf{p} \in \mathbb{R}^L, \sum_{l\in[L]} p_k = K, 0 \leq p_l \leq 1, \ \forall l \in [L]\right\}$. For $\mathbf{s} \in \mathbb{R}^L$, one can check that $\arg\operatorname{top}\Sigma_K(\mathbf{s}) = \arg\max_{\mathbf{p}\in\Delta^{L-1}_K} \langle \mathbf{p}, \mathbf{s}\rangle$. Recall that Z is a standard normal random vector, *i.e.*, $Z \sim \mathcal{N}(0, \mathrm{Id}_L)$.

- Δ_K^{L-1} is a convex polytope and the multivariate normal has positive differentiable density. So, we can apply [Ber+20, Proposition 2.2]. For that, it remains to determine the constant $R_{\Delta_K^{L-1}}$ and M_μ. First, $R_{\Delta_K^{L-1}} \triangleq \max_{\mathbf{p}\in\Delta_K^{L-1}} \|\mathbf{p}\|$. For simplicity, let us compute $\max_{\mathbf{p}\in\Delta_K^{L-1}} \|\mathbf{p}\|^2$, *i.e.*,

$$\max \|\mathbf{p}\|^2 \tag{4.18}$$
$$\textstyle\sum_{k\in[L]} p_k = K,\ \forall k \in [L], p_k \in [0,1]. \tag{4.19}$$

Note that this corresponds to the well known quadratic knapsack problem. A numerical solution can be obtained, see for instance [HKL80b]. To obtain our bound, note that for $\mathbf{p} \in \Delta_K^{L-1}$ one can check that

$$\|\mathbf{p}\|^2 = \sum_{k\in[L]} p_k^2 \leq \sum_{k\in[L]} p_k \quad (\text{since } \forall k \in [L], p_k \in [0,1]).$$

Hence, we have $\forall \mathbf{p} \in \Delta_K^{L-1}, \|\mathbf{p}\|^2 \leq K$. Now, one can check that this equality is achieved when choosing $\mathbf{p} = (1,\dots,1,0,\dots,0)^\top \in \mathbb{R}^L$ with K non-zeros values, yielding $\max_{\mathbf{p}\in\Delta_K^{L-1}} \|\mathbf{p}\| = \sqrt{K}$. Following [Ber+20, Proposition 2.2] this guarantees that $\mathrm{top}\Sigma_{K,\epsilon}$ is $\sqrt{K}$-Lipschitz.

Let us show now that $M_\mu \triangleq \sqrt{\mathbb{E}_Z[\|\nabla_Z \nu(Z)\|^2]} = \sqrt{L}$ with $\nu(Z) = \frac{1}{2}\|Z\|^2$ and $Z \sim \mathcal{N}(0, \mathrm{Id}_L)$. Hence, M_μ can be computed as:

$$\begin{aligned} M_\mu &= \sqrt{\mathbb{E}_Z\left[\|\nabla_Z \nu(Z)\|^2\right]} \\ &= \sqrt{\mathbb{E}_Z\left[\|Z\|^2\right]} \\ &= \sqrt{L}\ , \end{aligned}$$

where the last equality comes from $Z \sim \mathcal{N}(0, \mathrm{Id}_L)$. Following [Ber+20, Proposition 2.2] this guarantees that $\nabla\mathrm{top}\Sigma_{K,\epsilon}$ is $\frac{\sqrt{KL}}{\epsilon}$-Lipschitz.

- The last bullet of our proposition comes derives now directly from of [Ber+20, Proposition 2.3].

□

4.6.2.2 Proof of Proposition 4.3.5

Proof. • From the triangle inequality and proposition 4.3.3 we get for any $\mathbf{s}, \mathbf{s}' \in \mathbb{R}^L$:

$$\begin{aligned} \|\nabla\tau_{K,\epsilon}(\mathbf{s}) - \nabla\tau_{K,\epsilon}(\mathbf{s}')\| &= \|[\nabla\mathrm{top}\Sigma_{K,\epsilon}(\mathbf{s}) - \nabla\mathrm{top}\Sigma_{K,\epsilon}(\mathbf{s}')] \\ &\qquad - [\nabla\mathrm{top}\Sigma_{K-1,\epsilon}(\mathbf{s}) - \nabla\mathrm{top}\Sigma_{K-1,\epsilon}(\mathbf{s}')]\| \\ &\leq \|\nabla\mathrm{top}\Sigma_{K,\epsilon}(\mathbf{s}) - \nabla\mathrm{top}\Sigma_{K,\epsilon}(\mathbf{s}')\| \\ &\qquad + \|\nabla\mathrm{top}\Sigma_{K-1,\epsilon}(\mathbf{s}) - \nabla\mathrm{top}\Sigma_{K-1,\epsilon}(\mathbf{s}')\| \\ &\leq \left(2\frac{\sqrt{KL}}{\epsilon} + 2\frac{\sqrt{(K-1)L}}{\epsilon}\right)\|\mathbf{s}-\mathbf{s}'\| \leq 4\frac{\sqrt{KL}}{\epsilon}\|\mathbf{s}-\mathbf{s}'\| \end{aligned}$$

- Using the notation from [Ber+20, Appendix A], with $F(\mathbf{s}) = \mathrm{top}\Sigma_K(\mathbf{s})$ and $F_\epsilon(\mathbf{s}) = \mathrm{top}\Sigma_{K,\epsilon}(\mathbf{s})$, we get the following bounds:

$$0 \leq \mathrm{top}\Sigma_{K,\epsilon}(\mathbf{s}) - \mathrm{top}\Sigma_K(\mathbf{s}) \leq \epsilon \cdot \mathrm{top}\Sigma_{K,1}(\mathbf{0}) \tag{4.20}$$
$$0 \leq \mathrm{top}\Sigma_{K-1,\epsilon}(\mathbf{s}) - \mathrm{top}\Sigma_{K-1}(\mathbf{s}) \leq \epsilon \cdot \mathrm{top}\Sigma_{K-1,1}(\mathbf{0}) \ . \tag{4.21}$$

  Additionally, using the maximal inequality for *i.i.d.* Gaussian variables, see for instance [BLM13, Section 2.5], leads to:

$$\mathrm{top}\Sigma_{K,1}(\mathbf{0}) = \mathbb{E}\left[\sum_{k\in[K]} Z_{(k)}\right] \leq K\mathbb{E}\Big[Z_{(1)}\Big] \leq K\sqrt{2\log L}$$

  Subtracting (4.21) to (4.20), and reminding that $\tau_K(\mathbf{s}) = \mathrm{top}\Sigma_K(\mathbf{s}) - \mathrm{top}\Sigma_{K-1}(\mathbf{s})$ (and similarly $\tau_{K,\epsilon}(\mathbf{s}) = \mathrm{top}\Sigma_{K,\epsilon}(\mathbf{s}) - \mathrm{top}\Sigma_{K-1,\epsilon}(\mathbf{s})$) gives:

$$-\epsilon \cdot (K-1)\sqrt{2\log L} \leq \tau_{K,\epsilon}(\mathbf{s}) - \tau_K(\mathbf{s}) \leq \epsilon \cdot K\sqrt{2\log L} \ ,$$

  thus leading to:

$$|\tau_{K,\epsilon}(\mathbf{s}) - \tau_K(\mathbf{s})| \leq \epsilon \cdot C_{K,L} \ , \tag{4.22}$$

  with $C_{K,L} = K\sqrt{2\log L}$.

□

4.6.2.3 Proof of Proposition 4.3.7

Proof.

- First, note that $\mathbf{s} \mapsto \ell^{K,\epsilon}_{\text{Noised bal.}}(\mathbf{s}, y)$ is continuous as a composition and sum of continuous functions. It is differentiable wherever $\psi : \mathbf{s} \mapsto 1 + \tau_{K+1,\epsilon}(\mathbf{s}) - s_y$ is non-zero. From Definition 4.3.4 and Proposition 4.3.3 we get $\nabla_s \psi(\mathbf{s}) = \mathbb{E}[\arg \mathrm{top}\Sigma_{K+1}(\mathbf{s} + \epsilon Z)] - \delta_y$. The formula of the gradient follows from the chain rule.

□

4.6.3 Illustrations of the various losses encountered

In Figures 4.1 and 4.6, we provide a visualization of the loss landscapes for respectively $K = 1$ and $K = 2$ with $L = 3$ labels. With $L = 3$ labels, we display the visualization as level-sets restricted to a rescaled simplex: $2 \dots \Delta^2$. Moreover, we have min/max rescaled all the losses so that they fully range the interval $[0, 1]$. Note that as we are mainly interested in minimizing the losses, this post-processing would not modify the learned classifiers.

We provide also two additional figures illustrating the impact on our loss of the two main parameters: ϵ and M.

Table 4.7: *top-1 accuracy cifar100*

Label noise	ℓ^{K}_{CE}	$\ell^{5,1.0}_{\text{Smoothed Hinge}}$	$\ell^{5,0.2,10}_{\text{Noised bal.}}$
0.0	$\mathbf{76.6}\pm0.1$	69.2 ± 0.1	68.7 ± 0.3
0.1	71.0 ± 0.2	$\mathbf{71.2}\pm0.4$	68.3 ± 0.5
0.2	68.1 ± 0.1	$\mathbf{71.3}\pm0.3$	69.4 ± 0.5
0.3	65.5 ± 0.3	$\mathbf{70.8}\pm0.6$	69.3 ± 0.3
0.4	61.8 ± 0.4	$\mathbf{70.6}\pm0.2$	69.1 ± 0.4

Table 4.8: *regular top-K accuracy corresponding to the models in Table 4.5*

K	ℓ_{CE}	$\ell^{K,\tau}_{\text{Smoothed Hinge}}$	$\ell^{K,\epsilon,M}_{\text{Noised bal.}}$	focal	LDAM	$\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$
1	80.1 ± 0.1	79.8 ± 0.0	80.8 ± 0.1	79.8 ± 0.1	79.6 ± 0.1	$\mathbf{81.0}\pm0.1$
3	93.1 ± 0.0	93.2 ± 0.0	$\mathbf{93.5}\pm0.1$	93.5 ± 0.0	92.3 ± 0.1	93.5 ± 0.1
5	95.7 ± 0.0	95.0 ± 0.1	95.8 ± 0.0	$\mathbf{96.0}\pm0.0$	95.2 ± 0.2	95.8 ± 0.1
10	97.5 ± 0.0	95.5 ± 0.0	97.5 ± 0.0	$\mathbf{97.7}\pm0.0$	97.2 ± 0.1	97.5 ± 0.0

4.6.4 Additional experiments

CIFAR-100: Table 4.7 reports the top-1 accuracy obtained by the models corresponding to the results of Table 4.4. Hence, we show here a misspecified case: we optimized our balanced loss for $K = 5$, seeking to optimize top-5 accuracy, which is reported in Table 4.4, but report top-1 information. Table 4.7 shows that when there is no label noise, as expected cross entropy gives better top-1 accuracy than our top-5 loss. When the label noise is high, however, our loss leads to better top-1 accuracy.

Pl@ntNet-300K: Table 4.8 reports the top-K accuracy obtained by the models corresponding to the results of Table 4.5. The top-K accuracies are much higher than the macro-average top-K accuracies reported in Table 4.5 because of the long-tailed distribution of Pl@ntNet-300K. The models perform well on classes with a lot of examples which leads to high top-K accuracy. However, they struggle on classes with a small number of examples (which is the majority of classes, see [Gar+21]). Thus, for Pl@ntNet-300K top-K accuracy is not very relevant as it mainly reflects the performance of the few classes with a lot of images, ignoring the performance on challenging classes (the one with few labels) [Aff+17]. We report it for completeness and make a few comments: First, our balanced noise loss gives better top-K accuracies than the cross entropy or the loss from [BZK18] for all K. Then, $\ell^{K,\epsilon,M,m_y}_{\text{Noised imbal.}}$ and $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ produce the best top-1 accuracy, respectively 81.0 and 80.8.

However, we insist that in such an imbalanced setting the macro-average top-K accuracy reported in Table 4.5 is much more informative than regular top-K accuracy. We see significant differences between the losses in Table 4.5 which are hidden in Table 4.8 because of the extreme class imbalance.

ImageNet: We test $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ on ImageNet. We follow the same procedure as

Table 4.9: *ImageNet test top-K accuracy, ResNet-18.*

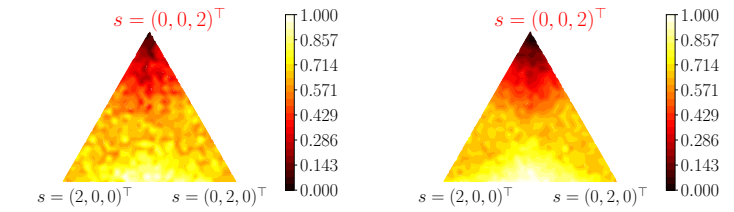
K	ℓ_{CE}	$\ell^{K,0.1}_{\text{Smoothed Hinge}}$	$\ell^{K,0.5,10}_{\text{Noised bal.}}$
1	72.24 ± 0.15	71.43 ± 0.14	$\mathbf{72.46} \pm 0.15$
5	90.60 ± 0.05	$\mathbf{90.71} \pm 0.06$	90.52 ± 0.07

[BZK18]: we train a ResNet-18 with SGD for 120 epochs with a batch size of 120 epochs. The learning rate is decayed by ten at epoch 30, 60 and 90. For $\ell^{K,\tau}_{\text{Smoothed Hinge}}$, the smoothing parameter τ is set to 0.1, the weight decay parameter to 0.000025 and the initial learning rate to 1.0, as in [BZK18]. For ℓ_{CE}, the weight decay parameter is set to 0.0001 and the initial learning rate to 0.1, following [BZK18]. For $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$, we use the same validation set as in [BZK18] to set ϵ to 0.5, the weight decay parameter to 0.00015 and the initial learning rate to 0.1. M is set to 10. We optimize all losses for $K = 1$ and $K = 5$. We perform early stopping based on best top-K accuracy on the validation set and report the results on the test set (the official validation set of ImageNet) in Table 4.9 (3 seeds, 95% confidence interval). In the presence of low label noise, with an important number of training examples per class and for a nearly balanced dataset, all three losses give similar results. This is in contrast with Table 4.4 and Table 4.5, where significant differences appear between the different losses in the context of label noise or class imbalance.

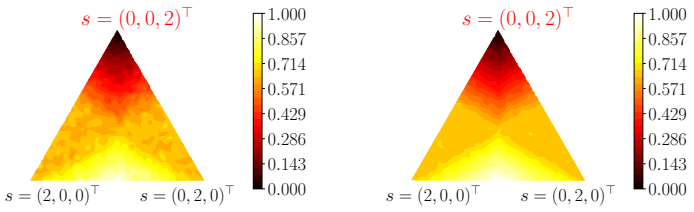
4.6.5 Hyperparameter tuning

Balanced case: For both experiments on CIFAR-100 and ImageNet, we follow the same learning strategy and use the same hyperparameters for $\ell^{K,\tau}_{\text{Smoothed Hinge}}$ than [BZK18]. For $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$, we refer the reader for the choice of ϵ and M respectively to Section 4.4.1 and Section 4.4.2: ϵ should be set to a sufficiently large value so that learning occurs and M should be set to a small value for computational efficiency.

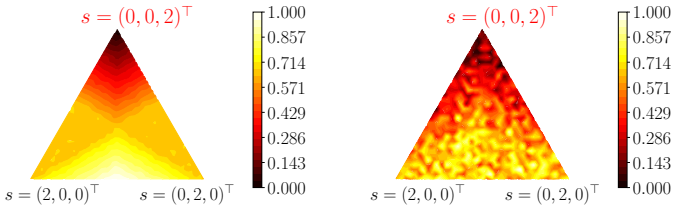
Imbalanced case: For our experiments on imbalanced datasets, we use the grid $\{0.5, 1.0, 2.0, 5.0\}$ for the parameter γ of the focal loss and $\{0.1, 1.0\}$ for the parameter τ of $\ell^{K,\tau}_{\text{Smoothed Hinge}}$. For $\ell^{\max m_y}_{\text{LDAM}}$ and $\ell^{K,\epsilon,M,\max m_y}_{\text{Noised imbal.}}$, the hyperparameter $\max m_y$ is searched in the grid $\{0.2, 0.3, 0.4, 0.5\}$ and we find in our experiments that the best working values of $\max m_y$ happen to be the same for both losses. For the scaling constant for the scores, we find that 30 and 50 are good default values for respectively $\ell^{\max m_y}_{\text{LDAM}}$ and $\ell^{K,\epsilon,M,\max m_y}_{\text{Noised imbal.}}$. Finally, for $\ell^{K,\epsilon,M,\max m_y}_{\text{Noised imbal.}}$, ϵ is searched in the grid $\{0.01, 0.05, 0.1\}$.



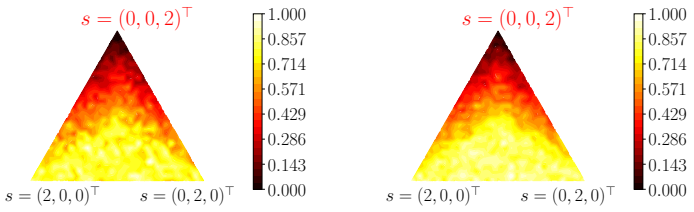
(a) $M = 1$ (b) $M = 5$



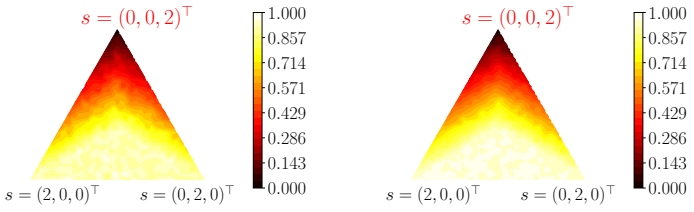
(c) $M = 10$ (d) $M = 50$



(e) $M = 100$ (f) $M = 1$

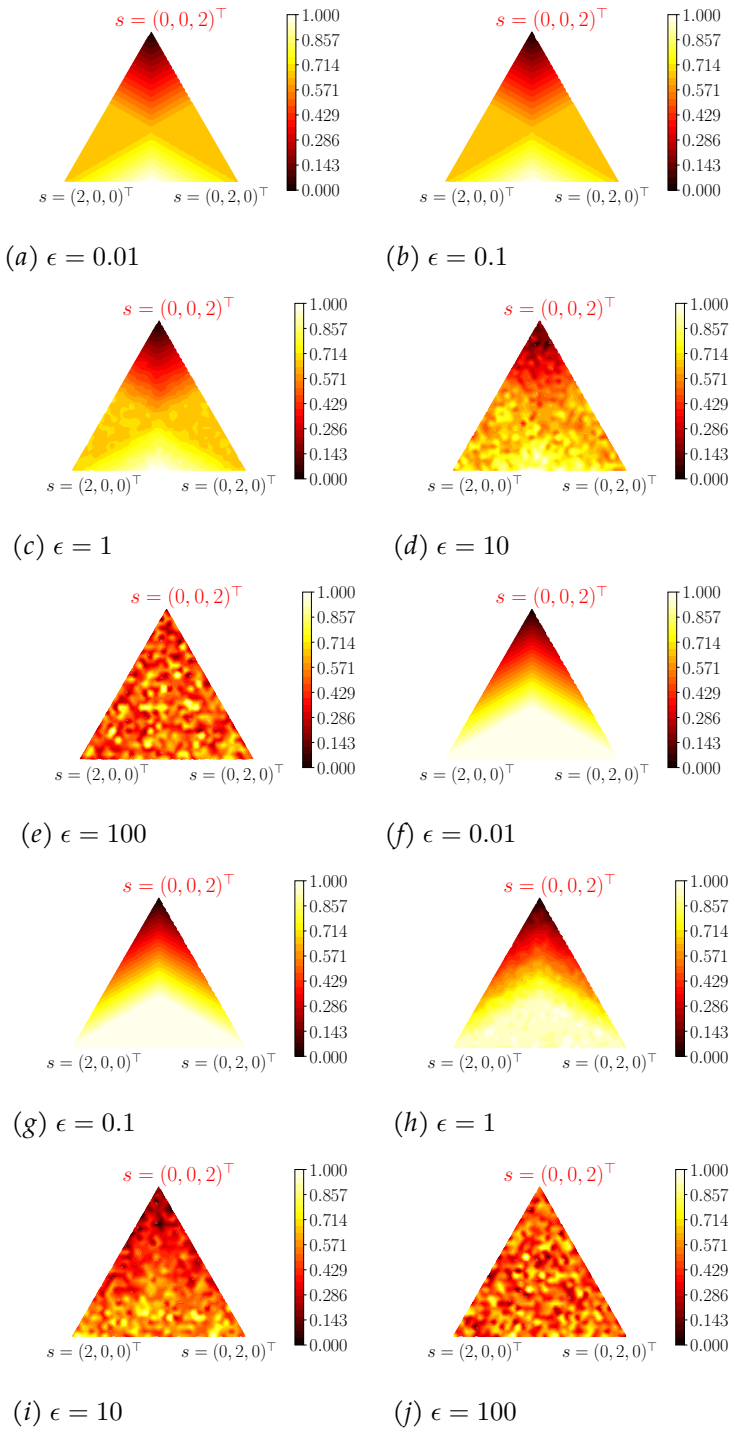


(g) $M = 5$ (h) $M = 10$



(i) $M = 50$ (j) $M = 100$

Figure 4.4: *Impact of the sampling parameter M on the loss $\ell^{K,1,M}_{\text{Noised bal.}}$ (top part: $K = 1$, bottom part: $K = 2$)*



(a) $\epsilon = 0.01$ (b) $\epsilon = 0.1$

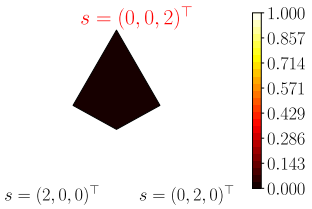
(c) $\epsilon = 1$ (d) $\epsilon = 10$

(e) $\epsilon = 100$ (f) $\epsilon = 0.01$

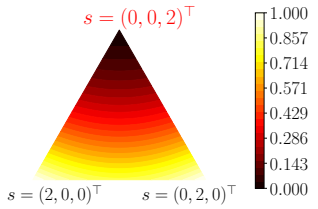
(g) $\epsilon = 0.1$ (h) $\epsilon = 1$

(i) $\epsilon = 10$ (j) $\epsilon = 100$

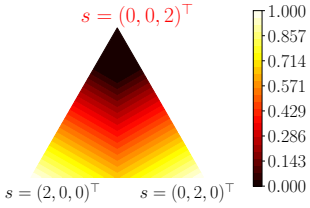
Figure 4.5: *Impact of the smoothing parameter ϵ on the loss $\ell^{K,\epsilon,50}_{\text{Noised bal.}}$ (top part: $K = 1$, bottom part: $K = 2$)*



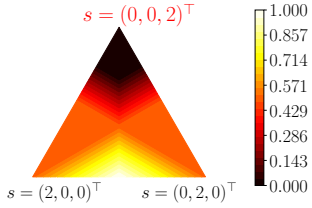
(a) *Top-K:* $\ell = \ell^K$.



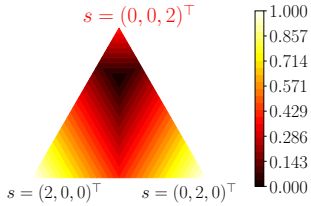
(b) *Cross-entropy:* $\ell = \ell_{CE}$.



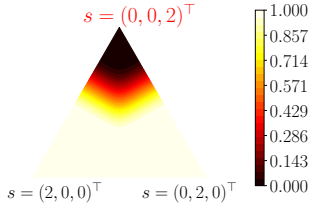
(c) *Multi-class hinge:* $\ell = \ell^K_{Hinge}$.



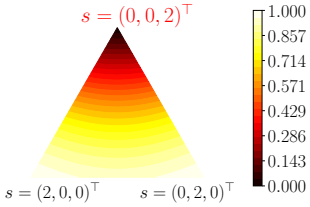
(d) *Calibrated hinge:* $\ell = \ell^K_{Cal.\ Hinge}$.



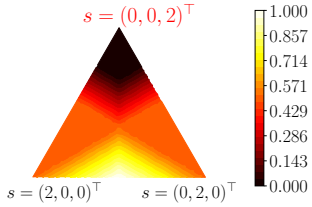
(e) *Convexified hinge:* $\ell = \ell^K_{CVXHinge}$.



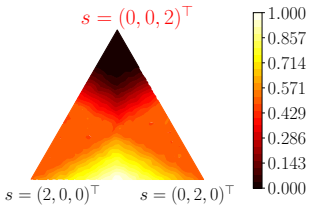
(f) *Smoothed hinge* $\ell^{K,0.1}_{Smoothed\ Hinge}$.



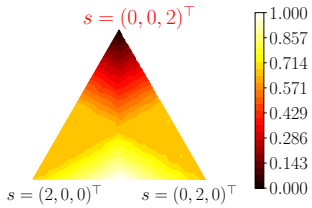
(g) *Smoothed hinge* $\ell^{K,1}_{Smoothed\ Hinge}$.



(h) *Noised balanced:* $\ell^{K,0.3,30}_{Noised\ bal.}$.



(i) *Noised balanced:* $\ell^{K,1,30}_{Noised\ bal.}$.



(j) *Noised imbalanced:* $\ell^{K,1,30,5}_{Noised\ imbal.}$.

Figure 4.6: *Level sets of the function* $\mathbf{s} \mapsto \ell(\mathbf{s}, y)$ *for different losses described in Table 4.1, for* $L = 3$ *classes,* $K = 1$ *and a true label* $y = 2$ *(corresponding to the upper corner of the triangles). For visualization the loss are rescaled between 0 and 1, and the level sets are restricted to vector* $\mathbf{s} \in 2 \cdot \Delta^2$. *The losses have been harmonized to display a margin equal to 1. For our proposed loss, we have averaged the level sets over 100 replications to avoid meshing artifacts.*

CHAPTER 5

AVERAGE-K LOSS FOR DEEP LEARNING

Contents

5.1 Introduction

The rise of visual sensors and the democratization of crowd-sourcing approaches (*e.g.,* citizen science) are contributing to the emergence of new massive image datasets with a very large number of classes [Hor+18; Gar+21; Yua+21; Rob+19]. These datasets contain many classes that are similar to each other and are prone to label ambiguity due to the crowd-sourced acquisition and/or annotation process. In such cases, it is difficult to achieve high levels of top-1 accuracy. This is consistent with theory: most of the time, the Bayes classifier has a non-zero error rate due to the random nature of the underlying generation process [DKD09b]. This is problematic for systems that aim to provide their users with a correct answer from an image [Har+23; Mac+21].

One way to deal with this difficulty is to allow the classifier to return more than one class. The most standard way to do this is through top-K classification, in which we allow the classifier to return exactly K candidate classes for all images [LHS15]. For instance, top-K accuracy is the official ImageNet [Rus+15] metric.

Although top-K classifiers reduce the error rate by returning K classes, they lack flexibility: for some clean, unambiguous images, it is not necessary to return K candidate classes. Conversely, for some ambiguous images, K may not be sufficient for the true class to be in the returned set. Figure 5.1 illustrates how ambiguity can vary from one image to the other.

To address this variability, classifiers that return a variable number of classes from an input must be used. There exists a broad range of set-valued classifiers [Chz+21] and strategies [FZV23], some more flexible than others. In this chapter, we focus on average-K classifiers [DH17] that return class sets of average size K, where the average is taken over the dataset. This constraint is less strict than for top-K classification, where the cardinal of the returned sets must be K for each instance. This flexibility allows returning more than K classes for ambiguous images and less than K classes for easy ones.

Controlling the average number of classes returned is useful for many applications, as it meets both UI (User Interface) design needs (*e.g.,* the average number of results should fit on a mobile app screen) and UX (User eXperience) design needs (*e.g.,* a recommender system should not recommend too many items on average). In some cases, the expected average size may also be known or estimated from other data sources (*e.g.,* the average number of species present at a given location).

The simplest approach to perform average-K classification is to threshold the softmax predictions of a deep neural network optimized with cross-entropy loss. While cross-entropy is theoretically grounded in the infinite sample limit [Lor20], no guarantee exists in the finite sample case.

In this chapter, we propose a novel method to optimize average-K accuracy by adding an auxiliary head to a classical deep learning model. One head is responsible for identifying candidate classes to be returned in the set and the second head maximizes the likelihood of each candidate class.

We experiment on two datasets and report significant improvements over vanilla

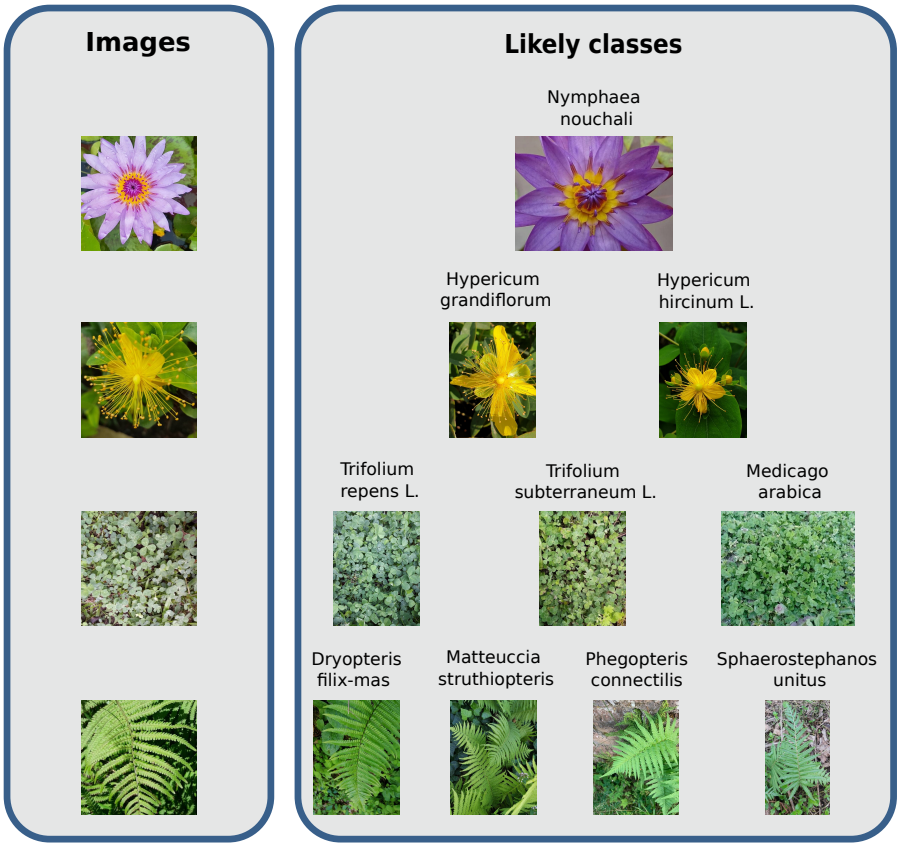


Figure 5.1: *Variety of ambiguity in Pl@ntNet-300K dataset* [Gar+21]. *The images on the left are from the dataset while the ones on the right represent the likely corresponding classes. The first row corresponds to the case where there is almost no uncertainty, the second row to the case where two classes are viable candidates for the image, etc.*

cross-entropy training and other methods.

5.2 Problem statement

We are in the multi-class setting, where the image/label pairs (X, Y) are assumed to be generated *i.i.d.* by an unknown probability distribution $\mathbb{P}$. Each image x is associated with a label $y \in [L] \coloneqq \{1, \ldots, L\}$, where L is the number of classes. Traditional multi-class classifiers are functions $h : \mathcal{X} \to [L]$ that map an image to a single label.

To reduce the risk of not returning the true class associated to an image, we are interested in set-valued classifiers $\Gamma : \mathcal{X} \to 2^{[L]}$ that map an image to a set of labels [Chz+21], where $2^{[L]}$ is the power set of $[L]$. Our objective is then to build a classifier g with minimal risk $\mathcal{R}(\Gamma) \coloneqq \mathbb{P}(Y \notin \Gamma(X))$.

A trivial solution would be to take $\Gamma(x) = [L]$ for all $x \in \mathcal{X}$, *i.e.*, the classifier that returns the set of all classes for each input image. To build a useful classifier, a constraint must be enforced on this optimization problem (as discussed in [Chz+21] where a unified framework encompassing several possible constraints is proposed). In this chapter, we focus on the average set size constraint that controls the average size of the

returned set:

$$\mathbb{E}_X[|\Gamma(X)|] \leq K \ \ , (\mathcal{C}) \tag{5.1}$$

where K is an integer and $|\cdot|$ denotes the cardinal.

It has been shown [DH17] that the optimal classifier Γ^* minimizing $\mathcal{R}$ while satisfying $\mathcal{C}$ is:

$$\Gamma^*(x) = \{j \in [L], \mathbb{P}(Y = j|X = x) \geq \lambda\} \ \ , \tag{5.2}$$

where $\lambda \in [0, 1]$ is calibrated so that K classes are returned on average.

These quantities are theoretical since we do not know $\mathbb{P}$. In practice, given a real dataset, the typical workflow for building an average-K classifier is to learn the model with cross-entropy on the training set and compute the threshold — so that on average K classes are returned — on the estimated conditional probabilities (softmax layer) with a validation set. The model is then evaluated on the test set.

While cross-entropy is theoretically grounded in the infinite limit sample case [Lor20], there is no guarantee in the finite sample regime. In this chapter, we propose an alternative to cross-entropy to optimize average-K accuracy by formulating the problem as a multi-label problem.

5.3 Related work

Several works have studied set-valued classification. The most studied case is top-K classification. In [LHS15], the authors propose a top-K hinge loss and optimize it with Stochastic Dual Coordinate Ascent [SSZ13]. In [YK20], the authors propose a slight variation of the top-K hinge loss with stronger theoretical guarantees. In [BZK18] and [Gar+22], the authors propose to smooth top-K hinge losses to make them suitable for deep learning.

Average-K classification is less studied. In [DH17], the authors derive the average-K Bayes classifier. They also show that minimizing a certain ϕ-risk with specific conditions on ϕ implies minimizing the average-K error. In [Lor20], the authors show that strongly proper losses are consistent for average-K classification, which means that minimizing the expected value of such a loss leads to minimizing the expected value of the average-K error. They also show that the cross-entropy loss is strongly proper, which gives a theoretical argument for using the cross-entropy loss for average-K classification. While the two previous works give theoretical results in the infinite limit case, we instead propose a practical method for optimizing average-K accuracy and show that it performs better than cross-entropy and other methods on real-world datasets.

Another framework close to set-valued classification is conformal prediction [VGS05; SV08; FZV23]. Conformal prediction also aims to generate prediction sets, but it relies on the use of calibration data to determine them based on hypothesis testing. It has the advantage of working on any pre-trained classifier $\hat{\Gamma} : \mathcal{X} \to [L]$, but

it does not optimize the model itself towards the set-valued classification objective. Furthermore, the availability of calibration data can be problematic, especially for classes with few training examples (often the most numerous in the case of real-world datasets [Gar+21]).

Set-valued classification can also be connected to multi-label classification [Dem+12; Liu+21]. Indeed, in both settings, a set of labels is predicted in output. A crucial difference, however, lies in the data generation process. In multi-label classification, each input x is associated with a set of labels, whereas in the set-valued setting, each input x is associated with a single label. The work of [Col+21] is the closest to ours. The authors study the positive-only multi-label classification case, which is a particular case of multi-label classification where a single positive class is observed for the training images, but all the other classes are not observed (meaning they could be either positive or negative, we do not know). Their objective is then to learn a good classifier with this partial information. Our setting is different in the sense that in multi-class classification a single object is present in the image, but we want to return a set of possible classes to reduce the risk. However, both settings share similarities: a single class label is available during training and for each input image we return a set of classes.

5.4 Set-valued classification as multi-label

5.4.1 Preliminaries

In most multi-class datasets [Gar+21; Kri09; Rus+15], the label y associated with an object o of true class y^* is estimated by the annotators from a partial representation of o: an image x. Figure 5.2 summarizes how the final label y is obtained in most cases.

It may be difficult to determine the true class y^* of an object o given the partial representation x. The image may be of poor quality, lack a discriminative feature of y^*, etc. In such cases, several candidate classes are plausible for the object o, which we denote by the set $\mathcal{S}(x) \in 2^{[L]}$, see Figure 5.1.

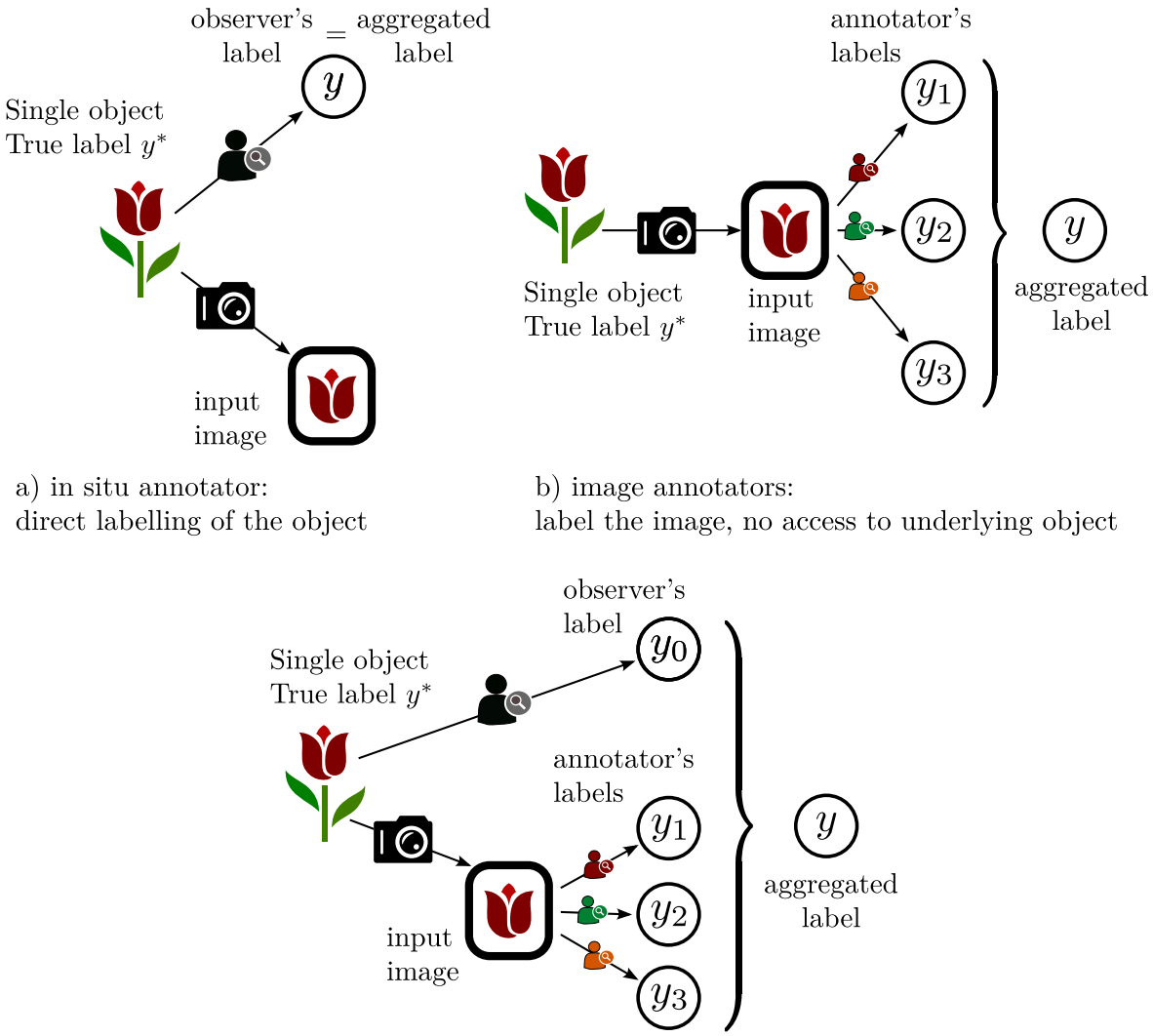
Regardless of this difficulty, a single label y —possibly different from y^*— is selected in multi-class datasets. In the case of multiple annotators, the default policy is to select the label with the most votes [Kha17].

In this chapter, we propose to dynamically estimate the sets $\mathcal{S}(x)$ during training and use $\mathcal{S}(x)$ as the ground truth labels with the binary cross-entropy loss in a multi-label fashion.

5.4.2 Notation

Let $\mathcal{N}_{train}$, $\mathcal{N}_{val}$ and $\mathcal{N}_{test}$ denote respectively the training set, the validation set and the test set indices. In the rest of the chapter, we use mini-batch gradient descent [Rud16b] to optimize all models. In the following, $B \subset \mathcal{N}_{train}$ denotes a random batch of training input.

For $i \in \mathcal{N}_{train} \cup \mathcal{N}_{val} \cup \mathcal{N}_{test}$, $\mathbf{z}_i \in \mathbb{R}^L$ denotes the score vector —logit— predicted by



a) in situ annotator:
direct labelling of the object

b) image annotators:
label the image, no access to underlying object

c) Mixed annotators:
mixed in-situ & image annotators

Figure 5.2: *Common annotation processes for multi-class datasets. The label y given to the input image is either obtained: a) after direct observation of the whole original object* e.g., [*Lit+20*] *b) by aggregating the labels of annotators having access only to a partial representation of the object (the input image),* e.g., [*Kri09; Rus+15*] *c) with a combination of both previous cases* e.g., [*Gar+21*]. *In all cases, annotations errors can occur resulting in a final label y different than the true class y^*.*

the model, z_{ij} its j-th component, and $y_i \in [L]$ the label assigned to i.

Given a batch $B = \{i_1, i_2, \dots, i_{|B|}\} \subset \mathcal{N}_{train}$, we define:

$$Z^B = \begin{bmatrix} — \mathbf{z}_{i_1} — \\ — \mathbf{z}_{i_2} — \\ \dots \\ — \mathbf{z}_{i_{|B|}} — \end{bmatrix} \quad \text{and} \quad Y^B = \begin{bmatrix} y_{i_1} \\ y_{i_2} \\ \dots \\ y_{i_{|B|}} \end{bmatrix},$$

respectively the batch predictions and batch labels.

Let P be a vector and $k \in \mathbb{N}^*$ a positive integer. $P_{[k]}$ will denote the k-th largest value of P.

Finally, let us note $\varsigma : \mathbb{R}^L \to \mathbb{R}^L$ the softmax function whose j-th component is given, for any $\mathbf{z}_i \in \mathbb{R}^L$ by:

$$\varsigma_j(\mathbf{z}_i) = \frac{e^{z_{ij}}}{\sum_{k=1}^{L} e^{z_{ik}}} \ ,$$

and σ the sigmoid function, defined for any $t \in \mathbb{R}$ by:

$$\sigma(t) = \frac{1}{1 + e^{-t}} \ .$$

5.4.3 Cross-entropy loss

Given a logit vector $\mathbf{z}_i \in \mathbb{R}^L$ and a label y_i, the cross-entropy loss for example i writes:

$$\ell_{\mathrm{CE}}(\mathbf{z}_i, y_i) = -\log(\varsigma_{y_i}(\mathbf{z}_i)) \ . \tag{5.3}$$

The partial derivatives read:

$$\frac{\partial \ell_{\mathrm{CE}}}{\partial z_j}(\mathbf{z}_i, y_i) = \begin{cases} \varsigma_{y_i}(\mathbf{z}_i)(1 - \varsigma_{y_i}(\mathbf{z}_i)), & \text{if } j = y_i \\ -\varsigma_{y_i}(\mathbf{z}_i) \cdot \varsigma_j(\mathbf{z}_i), & \text{o.w.} \end{cases} \ , \tag{5.4}$$

which is positive only if $j = y_i$.

Therefore, after a gradient descent update on ℓ_{CE}, z_{i,y_i} will increase and all other scores $(z_{ij})_{j \neq y_i}$ will decrease.

As stated in Sections 5.1 to 5.3, in the infinite limit case, ℓ_{CE} has theoretical grounds [Lor20]. However, it is not clear that this approach is optimal when only scarce/noisy data is available. Indeed, let us consider the case where two labels y_i and $\tilde{y}_i$ are equiprobable for the image x_i:

$$\mathbb{P}(Y = y_i | X = x_i) = \mathbb{P}(Y = \tilde{y}_i | X = x_i) = 0.5 \ ,$$

and assume that the label y_i was assigned to x_i when the dataset was constructed. With ℓ_{CE}, the score $z_{i,\tilde{y}_i}$ will decrease and z_{i,y_i} will increase during training, while we would like both $z_{i,\tilde{y}_i}$ and z_{i,y_i} to increase. Hence, in the context of high ambiguity, it is reasonable to formulate the problem as a multi-label classification task, in which each image x_i is associated with a set of labels $\mathcal{S}(x_i) \in 2^{[L]}$, the difficulty being that only one of them is observed, y_i.

5.4.4 Assume negative

A first multi-label approach similar to cross-entropy is to consider that for any $i \in \mathcal{N}_{train}$, $\mathcal{S}(x_i) = \{y_i\}$, *i.e.*, there is no ambiguity in the labels. This results in the Assume Negative

loss ℓ_{AN} [Col+21], essentially the binary cross entropy-loss with a single positive label:

$$\ell_{\text{AN}}(\mathbf{z}_i, y_i) = -\sum_{j=1}^{L} \Big[\mathbb{1}_{[j=y_i]} \log(\sigma(z_{ij})) + \tfrac{1}{L-1} \mathbb{1}_{[j\neq y_i]} \log(1-\sigma(z_{ij})) \Big] \ , \tag{5.5}$$

where the negative labels are weighted to have the same contribution as the positive label.

5.4.5 Expected positive regularization

The problem with ℓ_{AN} is that the second term of Equation (5.5) assumes that the scores of all classes different from y_i must be minimized, regardless of their relevance to example i. Removing the second term yields the positive cross-entropy loss [Col+21]:

$$\ell^{+}_{\text{BCE}}(\mathbf{z}_i, y_i) = -\sum_{j=1}^{L} \mathbb{1}_{[j=y_i]} \log(\sigma(z_{ij})) \ . \tag{5.6}$$

However, ℓ^{+}_{BCE} is minimized by predicting 1 for all classes, which is not desirable as it would lead to a large number of false positives.

A workaround proposed in [Col+21] is to constrain the current batch probability predictions $\sigma(Z^B)$ to sum to K on average, where σ is applied pointwise and the average is over the batch dimension. Here K is a hyperparameter that can be thought of as the expected average set size. More formally, the expected number of positive classes can be estimated as:

$$\hat{K}(Z^B) = \frac{1}{|B|} \sum_{i\in B} \sum_{j=1}^{L} \sigma(z_{ij}) \ . \tag{5.7}$$

The Expected positive regularization loss [Col+21] $\ell^{\beta}_{\text{EPR}}$ then reads:

$$\ell^{\beta}_{\text{EPR}}(Z^B, Y^B) = \tfrac{-1}{|B|} \sum_{i\in B} \log(\sigma(z_{i,y_i})) + \beta(\hat{K}(Z^B) - K)^2, \tag{5.8}$$

where β is a hyperparameter to be tuned on a validation set.

Although the idea behind $\ell^{\beta}_{\text{EPR}}$ seems reasonable, there is an infinite number of combinations for the matrix $\sigma(Z^B)$ to sum to K on average. In particular, the model could learn to place diffuse probabilities on all classes without promoting strong class candidate alternatives to the true label class in the training set.

5.4.6 Online estimation of labels

In [Col+21], the authors introduce a loss $\ell^{\lambda,\Theta}_{\text{ROLE}}$ that builds on $\ell^{\beta}_{\text{EPR}}$. In addition, they keep a matrix estimate of the unobserved labels $\Theta \in \mathbb{R}^{n_{train}\times L}$, where $n_{train} := |\mathcal{N}_{train}|$ is the number of training examples in the dataset. During training, the labels predicted

by the model are trained to match the current estimates of the labels in Θ via a cross-entropy term and, in addition, to satisfy the constraint in Equation (5.8). The role of the predicted and estimated labels is then reversed. For more details, we refer the reader to [Col+21].

Although $\ell^{\lambda,\Theta}_{\text{ROLE}}$ is more sophisticated than EPR, we find in our experiments that it does not perform well. Moreover, it requires tuning several hyperparameters and, most importantly, keeping in GPU memory a matrix of size $n_{train} \times L$, which is prohibitive for large datasets.

5.5 Proposed method

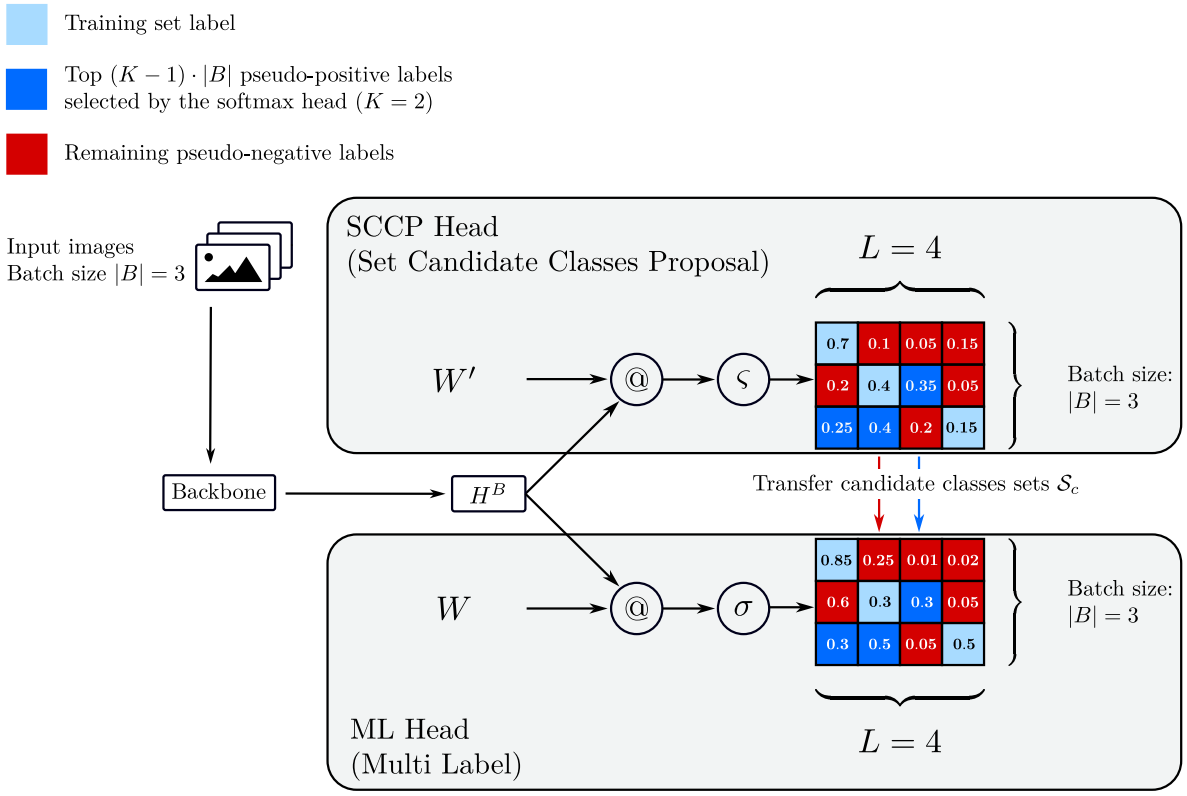


Figure 5.3: *The Set Candidate Classes Proposal -SCCP- head (on top) is responsible for determining, for each example of the batch, which classes to include in the target label set. The light blue cells correspond to the training set labels. Only one label is assigned per example/row. The dark blue cells correspond to the classes selected as pseudo-positives by the SCCP head. They correspond to the top $(K-1)|B|$ highest values of the SCCP's softmax prediction matrix deprived of the light blue true labels. In this example, $K=2$ and $|B|=3$, so $(2-1)\times 3 = 3$ classes are selected as pseudo-positives in the batch. They are assigned a pseudo-label 1 in the ML head. The remaining cells, the red ones, are those that were not selected as pseudo-positives by the SCCP head and are considered pseudo-negatives by the ML head (with a pseudo-label 0). Here @ denotes matrix multiplication.*

5.5.1 Outline

Let us assume that given a random batch $B \subset \mathcal{N}_{train}$ of examples, we are able to estimate for each $i \in B$ a set of possible classes different than y_i which we denote $\mathcal{S}_c(x_i)$. We can then define $\mathcal{S}(x_i) = \{y_i\} \cup \mathcal{S}_c(x_i)$. A legitimate objective is to increase the scores $(z_{ij})_{j \in \mathcal{S}(x_i)}$ and decrease the scores $(z_{ij})_{j \notin \mathcal{S}(x_i)}$, which is achieved by the following binary cross-entropy loss:

$$\begin{aligned} \ell(Z^B, Y^B, \mathcal{S}_c^B) = & -\frac{1}{|B|} \sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j=y_i]} \log(\sigma(z_{ij})) \\ & - \alpha \frac{\sum\limits_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \in \mathcal{S}_c(x_i)]} \log(\sigma(z_{ij}))}{\sum\limits_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \in \mathcal{S}_c(x_i)]}} \\ & - \alpha \frac{\sum\limits_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \notin \mathcal{S}(x_i)]} \log(1 - \sigma(z_{ij}))}{\sum\limits_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \notin \mathcal{S}(x_i)]}} \quad , \end{aligned} \tag{5.9}$$

where the hyperparameter $\alpha \in \mathbb{R}^+$ controls the weighting between the training set observed labels and the candidate labels and can be seen as setting a different learning rate for the "hypothetical" labels (*i.e.*, the pseudo-labels) and the observed labels. We used the notation $\mathcal{S}_c^B = \{\mathcal{S}_c(x_i), i \in B\}$. The main difficulty is: how to obtain the candidate labels $\mathcal{S}_c(x_i)$?

5.5.2 Candidate classes estimation

To this end, we propose a two-head model, where the first head is responsible for identifying the candidate classes $\mathcal{S}_c(x_i)$ for each $i \in B$ (Set Candidate Classes Proposal -SCCP- head) and the second head (Multi-Label -ML- head) optimizes its predictions with the loss from Equation (5.9) in a multi-label fashion with the candidate classes $\mathcal{S}_c(x_i)$ estimated by the SCCP head.

Let us denote $H^B \in \mathbb{R}^{B \times d}$ as the output of the second to last layer of a deep neural network, where d is the dimension of the latent space. We define each head prediction as the output of a single linear layer building on H^B: $Z^B = H^B W \in \mathbb{R}^{B \times L}$ for the BCE head and $Z'^B = H^B W' \in \mathbb{R}^{B \times L}$ for the SCCP head. To identify relevant candidate classes, we rely on cross-entropy for its theoretical foundations [Lor20]. More formally, we optimize the SCCP head with:

$$\ell_{\text{CE}}(Z'^B, Y^B) = -\frac{1}{|B|} \sum_{i \in B} \log(\varsigma_{y_i}(\mathbf{z}'_i)) \quad , \tag{5.10}$$

where $\mathbf{z}'_i \in \mathbb{R}^L$ is the score prediction of the SCCP head for example i.

To propose sets of candidate classes, we select the maximum activations of the matrix $[\Sigma_{ij}^{-\infty}]_{i\in B, j\in[L]}$ defined as:

$$\Sigma_{ij}^{-\infty} = \begin{cases} \varsigma_j(\mathbf{z}'_i), & \text{if } y_i \neq j \\ -\infty, & \text{otherwise} \end{cases} . \tag{5.11}$$

To construct the candidate classes sets $\mathcal{S}_c^B$, we then select the top $(K-1)|B|$ values of $\Sigma^{-\infty}$ to obtain sets of average size K. More formally, for $i \in B$, we define $\mathcal{S}_c(x_i)$ as:

$$\mathcal{S}_c(x_i) = \{j, \Sigma_{ij}^{-\infty} \text{ is in the top-}_{(K-1)|B|} \text{ values of } \Sigma^{-\infty}\} . \tag{5.12}$$

This choice leads to sets of average size K on the batch:

$$\begin{aligned} \frac{1}{|B|}\sum_{i\in B}|\mathcal{S}(x_i)| &= \frac{1}{|B|}\sum_{i\in B}|\mathcal{S}_c(x_i)\cup\{y_i\}| \\ &= \frac{(K-1)|B|+|B|}{|B|} \\ &= K \end{aligned} \tag{5.13}$$

An illustrative schema of the method is available in Figure 5.3. We can now plug the estimated candidate sets $\mathcal{S}_c^B$ into Equation (5.9):

$$\begin{aligned} \ell_{\text{BCE}}(Z^B, Y^B, \mathcal{S}_c^B) = &-\tfrac{1}{|B|}\sum_{\substack{i\in B\\ j\in[L]}} \mathbb{1}[j=y_i]\log(\sigma(z_{ij})) \\ &-\tfrac{\alpha}{(K-1)|B|}\sum_{\substack{i\in B\\ j\in[L]}} \mathbb{1}[j{\in}\mathcal{S}_c(x_i)]\log(\sigma(z_{ij})) \\ &-\tfrac{\alpha}{(L-K)|B|}\sum_{\substack{i\in B\\ j\in[L]}} \mathbb{1}[j{\notin}\mathcal{S}(x_i)]\log(1{-}\sigma(z_{ij})) \ , \end{aligned} \tag{5.14}$$

The model is then trained jointly by minimizing the sum of the two losses:

$$\ell_{\text{AVG-}}^{K,\alpha}(Z'^B, Z^B, Y^B) = \ell_{\text{CE}}(Z'^B, Y^B) + \ell_{\text{BCE}}(Z^B, Y^B, \mathcal{S}_c^B) \tag{5.15}$$

At test time, the SCCP head is no longer needed, so we simply use the predictions of the ML head for prediction.

5.5.3 Hyperparameters

Our method depends heavily on the batch size since the candidate classes are selected from the whole batch. If the batch size is one, then for all $i \in \mathcal{N}_{train}$ $|\mathcal{S}(x_i)| = K$ with $|\mathcal{S}_c(x_i)| = K-1$ and the method is not able to capture the variability of ambiguity. As soon as $|B| \geq 2$, the set sizes can vary within the batch, allowing to account for the difference in ambiguity between different images of the batch. In our experiments, we found that classical values of $|B|$ work well. The hyperparameter α should be tuned on a validation. We found that $[0.1, 10.0]$ is a good default search range. We include experiments in the supplementary material to study the influence of $|B|$ and α on average-K accuracy.

5.5.4 Discussion

Our method has the advantage of being both computationally and memory efficient since it only requires the addition of a linear layer. In particular, it does not require storing a matrix of size $n_{train} \times L$ in memory as in [Col+21], which is prohibitive for large datasets. Besides, instead of a constraint on the average value of the predictions [Col+21], which can be satisfied by the model in various ways, we dynamically infer the instances target sets with the first head.

Algorithm 1: Computation of the threshold λ_{val}.

Input : $n_{val} \coloneqq |\mathcal{N}_{val}|$, $X_{val} = \{x_i\}_{i \in \mathcal{N}_{val}}$, batch size $|B|$, model f_θ, K
Output: λ_{val}

Split X_{val} into $\lceil \frac{|B|}{n_{val}} \rceil$ batches $X_1, \ldots, X_{\lceil \frac{|B|}{n_{val}} \rceil}$
$f_1 \leftarrow f_\theta(X_1)$ `// get batch logits`
$P \leftarrow \varsigma(f_1)$ `// apply softmax`
for $i = 2$ **to** $\lceil \frac{|B|}{n_{val}} \rceil$ **do**
  $f_i \leftarrow f_\theta(X_i)$ `// get batch logits`
  $p_i \leftarrow \varsigma(f_i)$ `// apply softmax`
  $P \leftarrow \mathrm{CONCAT}(P, p_i)$ `// row axis concat.`
`/* At this point` P `is a` $n_{val} \times L$ `matrix */`
$P \leftarrow \mathrm{FLATTEN}(P)$ `// turn` P `into a vector`
$P \leftarrow \mathrm{SORT}(P)$ `// sort in decreasing order`
$\lambda_{val} \leftarrow \frac{1}{2}(P_{[Kn_{val}]} + P_{[Kn_{val}+1]})$
return λ_{val}

5.6 Experiments

5.6.1 Metrics

We compare $\ell^{K,\alpha}_{\mathrm{AVG}}$ with ℓ_{CE}, ℓ_{AN}, $\ell^{\beta}_{\mathrm{EPR}}$ and $\ell^{\lambda,\Theta}_{\mathrm{ROLE}}$ on two datasets with different degrees of ambiguity. We also include the balanced top-K loss $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ from Chapter 4 [Gar+22]. For all datasets, we follow the same workflow: we train a neural network on the dataset with the different methods we compare. Early stopping is performed on best validation

ℓ_{CE}	$\ell^{K,\alpha}_{\text{AVG-}}$	$\ell^{\lambda,\Theta}_{ROLE}$	ℓ_{AN}	ℓ^{β}_{EPR}	$\ell^{K,\epsilon,M}_{\text{Noised bal.}}$
96.83 ± 0.16	$\mathbf{97.35} \pm 0.06$	96.12 ± 0.11	96.71 ± 0.02	95.88 ± 0.05	96.32 ± 0.05

Table 5.1: *CIFAR-100 test average-5 accuracy,* (*DenseNet 40-40*)

average-K accuracy, computed as follows:

$$\textit{val avg-}K\textit{ accuracy} = \frac{1}{|\mathcal{N}_{val}|} \sum_{i \in \mathcal{N}_{val}} \mathbb{1}[\varsigma_{y_i}(z_i) \geq \lambda_{val}] \ , \tag{5.16}$$

where the computation of λ_{val} is described in Algorithm 1. We then report average-K accuracies on the test set, using the threshold λ_{val}:

$$\textit{test avg-}K\textit{ accuracy} = \frac{1}{|\mathcal{N}_{test}|} \sum_{i \in \mathcal{N}_{test}} \mathbb{1}[\varsigma_{y_i}(z_i) \geq \lambda_{val}] \ . \tag{5.17}$$

The hyperparameters specific to all tested methods are tuned on the validation set using grid search, and the best model is then evaluated on the test set. The results are the average of several runs with different seeds and reported with 95% confidence intervals.

5.6.2 CIFAR100

Training: We first experiment our method on CIFAR-100 [Kri09], a dataset with $L = 100$ classes. We split the original training set (50 000 images) into a balanced training set of 45 000 images and a balanced validation set of 5 000 images on which all hyperparameters are tuned. We train a DenseNet40-40 [Hua+17] for 300 epochs with SGD and a Nesterov momentum of 0.9, following [Gar+22; BZK18]. The batch size is set to 64 and the weight decay to 0.0001. The learning rate is initially set to 0.1 and divided by ten at epoch 150 and 225.

Results: We report test average-5 accuracy in Table 5.1. CIFAR-100 is composed of 20 superclasses each containing 5 classes, *e.g.*, the superclass "aquatic mammals" groups "beaver", "dolphin", "otter", "seal", and "whale". Therefore, most of the ambiguity resides within each superclass, and we are able to achieve high average-5 accuracies ($\sim$ 96-97%, cf. Table 5.1.) This relatively low ambiguity explains the good performances of ℓ_{CE} and ℓ_{AN}. We find that ℓ^{β}_{EPR} and $\ell^{\lambda,\Theta}_{ROLE}$ lag behind, while $\ell^{K,\alpha}_{\mathrm{AVG}}$ based on the proposal of candidate sets benefits from a performance gain over all the other methods.

5.6.3 Pl@ntNet-300K

Description: Pl@ntNet-300K [Gar+21] is a plant image dataset of 306 146 images and 1 081 classes (species) that contains a lot of class ambiguity. It is composed of 303 *genera* (superclasses) each comprising one or more species. Ambiguity is particularly important within a *genus*: for instance, two orchid species may be very similar. In Pl@ntNet-300K, a *genus* can include from one to several dozen species. This makes the ambiguity in this dataset very variable (see Figure 5.1). Moreover, the labels are crowdsourced so Pl@ntNet-300K is particularly prone to label noise. These reasons make it a perfect candidate for average-K classification.

Training: We finetune a ResNet-18 [He+16] pre-trained on ImageNet [Rus+15] for 30 epochs. We use SGD and a Nesterov momentum of 0.9, with a batch size of 32 and

K	2	3
ℓ_{CE}	89.63 ± 0.08 (**27.08**/**65.83**/85.97)	92.64 ± 0.17 (38.44/75.55/90.50)
$\ell^{K,\epsilon,M}_{\text{Noised bal.}}$	**90.48** ± 0.05 (24.98/63.08/**87.21**)	93.60 ± 0.09 (38.46/73.67/91.85)
$\ell^{K,\alpha}_{\text{AVG}}$	90.34 ± 0.06 (23.77/61.81/86.74)	**93.81** ± 0.10 (**40.39**/**76.50**/**92.17**)
ℓ_{AN}	85.41 ± 0.15 (2.05/30.26/76.64)	90.15 ± 0.17 (7.46/47.20/86.04)
$\ell^{\beta}_{\text{EPR}}$	86.30 ± 0.17 (9.01/37.27/77.72)	90.49 ± 0.14 (18.49/51.58/85.19)
K	5	10
ℓ_{CE}	95.11 ± 0.18 (35.39/83.65/94.59)	97.11 ± 0.09 (46.58/91.71/97.30)
$\ell^{K,\epsilon,M}_{\text{Noised bal.}}$	95.75 ± 0.07 (49.90/81.39/95.00)	97.26 ± 0.03 (56.49/88.78/97.26)
$\ell^{K,\alpha}_{\text{AVG}}$	**96.42** ± 0.09 (**55.83**/**88.19**/**95.90**)	**98.23** ± 0.03 (**75.75**/**94.14**/**98.08**)
ℓ_{AN}	93.88 ± 0.12 (20.31/67.55/92.66)	96.86 ± 0.06 (42.61/85.49/96.87)
$\ell^{\beta}_{\text{EPR}}$	93.63 ± 0.04 (31.02/65.85/90.79)	95.99 ± 0.04 (41.44/78.61/95.09)

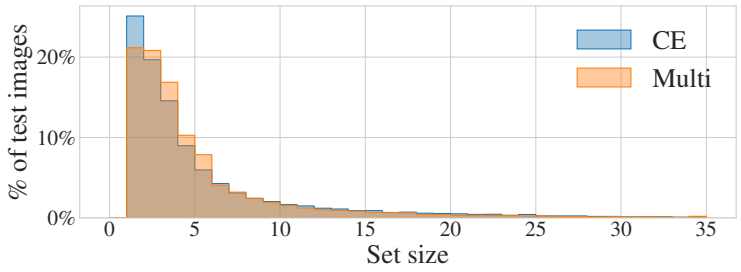
Table 5.2: *Pl@ntNet-300K test average-K accuracy* (*ResNet-18*). *The three numbers in parentheses represent respectively the mean average-K accuracies of 1*) *few shot classes* (< 20 *training images*) *2*) *medium shot classes* ($20 \leq \cdot \leq 100$ *training images*) *3*) *many shot classes* (> 100 *training images*).

an initial learning rate of 0.002, divided by ten at epoch 25. The weight decay is fixed at 0.0001. All methods are trained for $K \in \{2, 3, 5, 10\}$ and the average-K accuracy on the test set is then reported for each value of K.

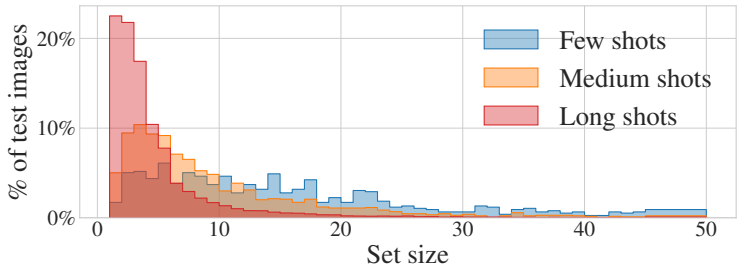
Results and interpretation: The results can be found in Table 5.2. It shows that our method is more effective for all K except when $K = 2$ where it gives results similar to the top-K loss. For high values of K, the gain is particularly important for few-shot and medium-shot classes, *i.e.*, classes with few examples (for a precise definition of few/medium/long shot classes, see the caption of Table 5.2). For instance, for $K = 10$, the average-10 accuracy of few-shot classes is 75.75 for our method compared to 56.49 for the top-K loss and 46.58 for the cross-entropy. These results are interesting because of Pl@ntNet-300K's long-tail: few-shot and medium-shot classes account for 48% and 25% of the total number of classes, respectively. This means that the model recognition capabilities are significantly higher for a vast majority of the classes. It should be noted that high values of K can arise in certain applications, for instance for the diagnosis of rare diseases [Fav+20] or for the automatic prediction of likely species at a particular location [Lor+22].

From Table 5.2 we see that the most naive methods ℓ_{AN} and $\ell^{\beta}_{\text{EPR}}$ perform poorly compared to the other losses. In particular, ℓ_{AN} which gave decent performances on CIFAR-100 struggles on Pl@ntNet-300K. One possible reason is that ℓ_{AN} assumes no class ambiguity while Pl@ntNet-300K has many.

It is worth noting that $\ell^{K,\epsilon,M}_{\text{Noised bal.}}$ gives interesting results and even outperforms cross-entropy. This is not surprising since the top-K loss optimizes top-K accuracy, and average-K and top-K classifiers are somewhat related (top-K classifiers are particular average-K classifiers that return K classes for each example). For a thorough comparison of top-K and average-K classification, we refer the reader to [Lor20].



(*a*) *Histogram of set sizes for Pl@ntNet-300K test set images for a model trained with* ℓ_{CE} *or* $\ell^{5,\alpha}_{\mathrm{AVG}}$ *to return sets of size 5 on average* (*ResNet-18*)



(*b*) *Histogram of set sizes for Pl@ntNet-300K test set images by category* (*few-shot, medium shot, long-shot classes*) *for a model trained with* $\ell^{5,\alpha}_{\mathrm{AVG}}$ *to return sets of size 5 on average* (*ResNet-18*)

Figure 5.4: *Histograms of set sizes.*

Distribution of set sizes: Figure 5.4(a) shows the repartition of set sizes for Pl@ntNet-300K test images, for a model trained with $\ell^{5,\alpha}_{\mathrm{AVG}}$ or ℓ_{CE} to return sets of average size 5. The cross-entropy is over-confident for many images, which results in an important number of sets of size one, whereas our method is more conservative and tends to return fewer sets of size one but more sets of size two, three and four.

Figure 5.4(b) shows the distribution of set size for few-shot, medium-shot and long-shot classes for a model trained with $\ell^{5,\alpha}_{\mathrm{AVG}}$ to return sets of average size 5. It appears clearly that images belonging to long-shot classes are associated with small set sizes. This is because the model saw enough training images belonging to these classes to make confident predictions. For medium-shot classes, the mode of the distribution is higher and the tail is heavier. For the most challenging images that belong to few-shot classes, the uncertainty results in larger set sizes, going up to ~ 50 classes.

5.7 Limitations and future work

While $\ell^{K,\alpha}_{\mathrm{AVG}}$ allows practical gains in average-K accuracy, its particular structure, based on a two-headed deep neural network makes its theoretical analysis difficult. In particular, it can not be shown easily that $\ell^{K,\alpha}_{\mathrm{AVG}}$ is average-K calibrated [Lor20] like the cross-entropy.

We have proposed a loss function for average-K classification which is a particular instance of set-valued classification. Other settings exist [Chz+21] (point-wise error control, average coverage control) and could be the object of ad-hoc optimization methods in future work.

5.8 Conclusion

We propose a loss function to optimize average-K accuracy, a setting in which sets of variable size are returned by the classifier to reduce the risk. Our method is based on the addition of an auxiliary head in the deep neural network trained the cross-entropy whose goal is to propose candidate class sets for the current batch of images. The candidate classes identified by the auxiliary head are then treated as pseudo-positives by a multi-label head optimized with the binary cross entropy. We show that our method compares favorably to the cross-entropy loss and other binary methods as well as to a top-K loss. We further show that the gain in average-K accuracy increases with K and is substantial for classes in the tail of a heavily imbalanced dataset. Our method has the advantage to be both memory and computationally efficient since it estimates the candidate classes on the fly with a single linear layer.

5.9 Appendix

5.9.1 Hyperparameter sensitivity

We study the impact of the hyperparameters α and $|B|$ on average-K accuracy by conducting further experiments on CIFAR-100 (Section 5.6.2).

Figure 5.5 shows how CIFAR-100 average-5 accuracy varies as a function of the hyperparameter α. Average-5 accuracy is stable over a wide range of α values (roughly 10^{-2} to 10^{1}), which means that α does not require a precise tuning to obtain good results. It drops sharply for high α values, *i.e.*, when the candidate classes have much more weight than the annotated labels of the training set in the objective function.

Figure 5.6 shows how average-K accuracy varies with the batch size for a model trained with ℓ_{CE} or $\ell_{\mathrm{AVG}}^{K,\alpha}$. For a fair comparison we maintain the ratio of learning rate to batch size constant. As expected, average-K accuracy decreases for both methods when the batch size becomes too small. However, we find that $\ell_{\mathrm{AVG}}^{K,\alpha}$ is more robust than ℓ_{CE} to large batch size values. This can be explained by the choice of more relevant candidate classes when the batch size becomes large. This is counterbalanced by the empricical fact that SGD tends not to work well with very large batch sizes in deep learning.

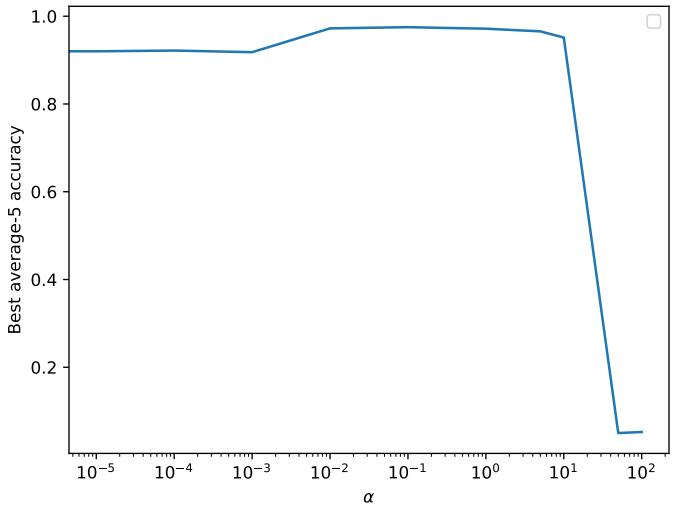


Figure 5.5: *CIFAR-100 best validation average-5 accuracy for a DenseNet 40-40 trained with $\ell_{\mathrm{AVG}}^{5,\alpha}$ for different values of α.*

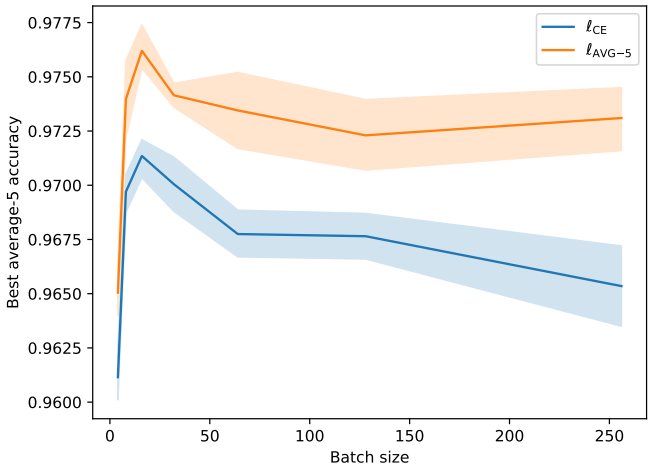


Figure 5.6: *CIFAR-100 best validation average-5 accuracy as a function of batch size for a DenseNet 40-40 trained with $\ell^{5,\alpha}_{\text{AVG}}$ or ℓ_{CE}. For a fair comparison we maintain the ratio of learning rate to batch size constant. The 95% confidence interval is represented.*

5.9.2 Experiments details

We report in Tables 5.3 and 5.4 the hyperparameters selected after grid search for all losses, for CIFAR-100 and Pl@ntNet-300K datasets respectively.

loss	hyperparameters
ℓ_{CE}	-
$\ell_{\text{AVG-K}}$	$\alpha = 0.3$
ℓ_{AN}	-
ℓ^{β}_{EPR}	$\beta = 0.01$
$\ell^{K,\epsilon,M}_{\text{Noised bal.}}$	$\epsilon = 0.2$
$\ell^{\lambda,\Theta}_{ROLE}$	$\lambda = 0.0$, learning rate Θ: $\times 1.0$

Table 5.3: *Hyperparameters selected after grid search for the CIFAR-100 experiments.*

loss	hyperparameters
ℓ_{CE}	-
$\ell_{\text{AVG-K}}$	$\alpha = 5.0$
ℓ_{AN}	-
ℓ^{β}_{EPR}	$\beta = 0.001$
$\ell^{K,\epsilon,M}_{\text{Noised bal.}}$	$\epsilon = 1.0$

Table 5.4: *Hyperparameters selected after grid search for the Pl@ntNet-300K experiments.*

CHAPTER 6

CONCLUSION AND PERSPECTIVES

Contents

6.1 Summary and possible lines of work

In this manuscript, we have proposed two methods to optimize respectively top-K and average-K accuracy. Our methods have proved efficient on datasets with ambiguous classes and a long-tailed distribution. The top-K loss we proposed was based on the smoothing of the top-K function with the perturbed optimizer framework. For the average-K loss, we introduced a two-headed model, where one head is responsible for identifying candidate classes to include in the set of classes and the other head is optimized with a binary cross entropy loss with the candidate classes as ground truth.

Overall, we have seen in Chapter 4 and 5 that it is possible to design loss functions that perform better than cross-entropy in the finite sample regime. We have seen that the ad-hoc methods we proposed are particularly well suited for datasets with high ambiguity and a long-tailed distribution, which arises in many real-world situations. In such cases, we obtain significant gain over cross-entropy, especially so on the few-shot classes, see for instance Table 4.4, Table 4.5 and Table 5.2. When data is abundant and numerous, cross-entropy works as well as our methods, as hinted by results in Table 4.9. This is not surprising as we have seen that cross-entropy is consistent for both top-K and average-K classification, making it a solid default choice.

While we have made two propositions for the top-K and average-K settings, there are other formulations as we have seen in Chapter 2, namely the average error and point-wise error settings. These methods attempt to minimize the average size while imposing a constraint respectively on the average error and point-wise error. We have not delved into these methods because they do not provide guarantees on the set size. In particular, both settings could lead to arbitrarily large sets, which is impractical in the context of Pl@ntNet, where one desires to return only a few species to the user. In contrast, top-K and average-K classification provide a certificate on the set size, either point-wise or on average. However, it would be interesting to derive methods for these formulations as well. This could be the object of future work.

Finally, some difficult theoretical work remains. It would be valuable to produce a theoretical analysis of the finite case to better understand the differences between the proposed approaches and cross-entropy. In particular, it would be interesting to understand what is happening on the long-tail classes and why we get large differences in performance on these classes.

6.2 Towards a sparse multi-label loss

As we have mentioned, multi-label classification is different from set-valued classification. The difference lies in the label space: in multi-label classification, an input example x can be associated to several labels: $\mathcal{Y} = 2^{[L]}$. In set-valued classification, a single label is associated to the input x. While this manuscript only dealt with set-valued classification, multi-label classification is very common as in many cases several classes can appear in an image. For instance, in a x-ray image classification [Bus+20], you can have several pathologies. In genomics, sequences often have multiple functional annotations [ZZ06], etc. Inspired by insights from top-K classification, we present

here ongoing work on a multi-label loss based on the projection on the top-K simplex.

In this section, we are in the multi-label classification setting *i.e.*, $\mathcal{Y} = 2^{[L]}$: several labels may be associated with an image x. In the rest of this section, $K \in [L]$. We recall that the simplex is denoted by:

$$\Delta^{L-1} \coloneqq \{\mathbf{p} \in \mathbb{R}^L, \mathbf{p} \geq 0, \sum_{i=1}^{L} p_i = 1\} \ . \tag{6.1}$$

Let us define the top-K simplex:

$$\Delta_K^{L-1} \coloneqq \{\mathbf{q} \in \mathbb{R}^L, 0 \leq \mathbf{q} \leq 1, \sum_{i=1}^{L} q_i = K\} \ . \tag{6.2}$$

The top-K simplex is the space of vectors that sum to K but where each entry is comprised between 0 and 1. Notice that the simplex and top-K simplex coincide when $K = 1$: $\Delta^{L-1} = \Delta_1^{L-1}$.

6.2.1 Motivation

The most standard loss for multi-label classification is the binary cross-entropy loss, which writes:

$$\ell_{BCE}(\mathbf{s}, y) = \sum_{j=1}^{L} -y \log(\sigma(s_i)) - (1-y) \log(1 - \sigma(s_i)) \ , \tag{6.3}$$

where σ denotes the sigmoid function.

This loss is theoretically grounded since it is consistent for multi-label classification. One drawback of this approach is that it produces a prediction with full support, *i.e.*, the predicted probabilities $\sigma(s_i)$ are non-zero. However, in most multi-label classifications problems, the number of classes present in the input is typically very small compared to the total number of classes. This motivates the need for a sparse multi-label loss, which we introduce next.

6.2.2 From `sparsemax` to `sparse-top`

We recall that the softmax can be seen as a projection onto the simplex:

$$\texttt{softmax}(\mathbf{s}) = \underset{\mathbf{p} \in \Delta^{L-1}}{\arg\min} -\langle \mathbf{s}, \mathbf{p} \rangle - H(\mathbf{p}) \ , \tag{6.4}$$

where $H(\mathbf{p}) \coloneqq -\sum_{j=1}^{L} p_j \log(p_j)$ is the entropy function. This simplifies into the most famous formulation for the `softmax`:

$$\texttt{softmax}(\mathbf{s})_j = \frac{\exp(s_j)}{\sum_{l=1}^{L} \exp(s_l)} \ . \tag{6.5}$$

From equation (6.5) it is clear that the `softmax` has full support, *i.e.*, $\texttt{softmax}(\mathbf{s})_j > 0$ for all $j \in [L]$. For some applications, a sparse distribution is desired. To tackle this,

the `sparsemax` [MA16] was introduced as a substitution for `softmax`. We recall its definition:

$$\texttt{sparsemax}(\mathbf{s}) = \underset{\mathbf{p}\in\Delta^{L-1}}{\arg\min} \|\mathbf{s}-\mathbf{p}\|^2 \quad . \tag{6.6}$$

It consists simply of the orthogonal projection on the simplex. The authors have shown that equation (6.6) leads to the following sparse solution:

$$\texttt{sparsemax}(\mathbf{s})_j = \max(0, s_j - \tau(\mathbf{s})) \quad , \tag{6.7}$$

where $\tau : \mathbb{R}^L \to \mathbb{R}$ is the unique function that satisfies, for all $\mathbf{s} \in \mathbb{R}^L$:

$$\sum_{j=1}^{L} \max(0, s_j - \tau(\mathbf{s})) = 1 \quad . \tag{6.8}$$

From (6.7) is is clear that `sparsemax` only has some components activated while the rest are zero. `sparsemax` is obtained by simple thresholding of the logits, where components below the threshold are zeroed out and where the threshold is chosen so that the sum of the "excess" $\sum_{j=1}^{L} \max(0, s_j - \tau(\mathbf{s}))$ is equal to one.

As `softmax` is used for the cross-entropy loss, similarly, we can design a loss function with `sparsemax`. We recall that the cross-entropy loss is defined by:

$$\ell_{\text{CE}}(\mathbf{s}, y) = -\log(\text{softmax}_y(\mathbf{s})) \quad , \tag{6.9}$$

whose gradient reads:

$$\nabla_{\mathbf{s}} \ell_{\text{CE}}(\mathbf{s}, y) = -\delta_y + \text{softmax}(\mathbf{s}) \quad , \tag{6.10}$$

where δ_y is the vector with one at coordinate y and 0 otherwise. By analogy to (6.10), the authors of `sparsemax` have designed a loss whose gradient is equal to:

$$\nabla_{\mathbf{s}} \ell_{\text{sparsemax}}(\mathbf{s}, y) = -\delta_y + \text{sparsemax}(\mathbf{s}) \quad , \tag{6.11}$$

which by integration yields:

$$\ell_{\text{sparsemax}}(\mathbf{s}, y) = -s_y + \frac{1}{2} \sum_{j \in S(\mathbf{s})} (s_j^2 - \tau(\mathbf{s})^2) + \frac{1}{2} \quad , \tag{6.12}$$

where $S(\mathbf{s}) :== \{j, \text{sparsemax}_j(\mathbf{s}) > 0\}$. The sparsemax is not well suited for multi-label classification. Indeed, it produces a vector $\mathbf{p}$ in the simplex, which is appropriate for multiclass classification, but what if we want to predict several labels? Actually, the authors proposed as an application of the sparsemax the task of proportion estimation, where the target is a proportion $\mathbf{q} \in \Delta^{L-1}$. This problem can occur in an agronomic context for instance, when you have to estimate grain proportion. But in multilabel classification, the target $y \in \{0, 1\}^L$ does not belong to the simplex, it is a vector of zeros and ones. The sparsemax can not predict such a distribution as it sums to one. What could be done is to normalize such labels so that they belong to the simplex. For

instance, a target vector $y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ could be transformed to $y = \begin{bmatrix} 0 \\ \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix}$. This is a bit odd, as this would lead to stronger signal for examples with few classes and a more diffuse response for examples with many classes. Instead, we would like to directly predict the $0/1$ target vector. For this purpose, we introduce `sparse-top` as:

$$\texttt{sparse-top}(K, \mathbf{s}) \coloneqq \underset{\mathbf{q} \in \Delta_K^{L-1}}{\arg\min} \|\mathbf{s} - \mathbf{q}\|^2 \tag{6.13}$$

6.2.3 Computation of `sparse-top`

Equation (6.13) actually belongs to a family of problems known as continuous quadratic knapsack [HKL80a]. This type of problems can be found in many applications such as resource allocation [BH81; HH95], multicommodity network flow problems [Ali+80], traffic equilibrium problems [DS69], etc.

As in the case $K = 1$, equation (6.13) admits a rather simple expression in the general case $K \geq 1$:

$$\texttt{sparse-top}_j(K, \mathbf{s}) = \max(0, \min(1, s_j - \lambda_K(\mathbf{s}))) \ , \tag{6.14}$$

where $\lambda_K : \mathbb{R}^L \to [0, 1]$ is the threshold function that satisfies:

$$\sum_{i=1}^{L} \texttt{sparse-top}_i(K, \mathbf{s}) = K \ . \tag{6.15}$$

Equation (6.14) ressembles the particular $K = 1$ case of Equation (6.7). Again the solution is obtained by thresholding the logits. The difference is that this time, the solution is truncated at 1 and the sum of the "excess" (the difference between the logit and the threshold) must be equal to K. Equation (6.14) induces three disjoint sets: the coordinates whose projections is equal to one, the coordinates whose projections is equal to zero and the coordinates whose projections lies strictly between 0 and 1. Therefore we define:

$$\begin{aligned} H_K(\mathbf{s}) &= \{i \in [L], \texttt{sparse-top}_i(K, \mathbf{s}) = 1\} \\ M_K(\mathbf{s}) &= \{i \in [L], 0 < \texttt{sparse-top}_i(K, \mathbf{s}) < 1\} \\ L_K(\mathbf{s}) &= \{i \in [L], \texttt{sparse-top}_i(K, \mathbf{s}) = 0\} \ . \end{aligned}$$

The main difficulty in computing `sparse-top` for a given input vector $\mathbf{s}$ comes down to estimate the threshold $\lambda_K(\mathbf{s})$. Several algorithms have been developed to address this. They can be grouped into two groups: breakpoint searching methods [HKL80a; HH95; CH94; CM87; Bru84; MPJ89; PK90; Mac+03; Kiw07] and variable fixing methods [Kiw08; RJL92; Mic86]. For breaking points methods, there are algorithms with complexity $O(L)$ to find the threshold and therefore the whole $\texttt{sparse-top}_K$ is of complexity $O(L)$, where L is the size of the input vector.

6.2.4 Derivatives

In deep learning, it is essential to provide the jacobian of the functions we use to train the model end to end. In this section, we provide the jacobian of `sparse-top` and compare it with that of the `softmax` and `sparsemax`. First, note that from Equation (6.14) that `sparse-top` is differentiable everywhere except at the transition points between the sets H_K, M_K and L_K. We remind the derivatives of the `softmax`:

$$\frac{\partial \texttt{softmax}_i}{\partial s_j}(\mathbf{s}) = \texttt{softmax}_i(\mathbf{s})(\delta_{ij} - \texttt{softmax}_j(\mathbf{s})) \ . \tag{6.16}$$

To compute the derivatives of `sparse-top`, we must first compute the derivatives of $\mathbf{s} \to \lambda_K(\mathbf{s})$. We have:

$$\frac{\partial \lambda_K}{\partial s_j}(\mathbf{s}) = \frac{1}{|M_K(\mathbf{s})|} \text{ if } j \in M_K(\mathbf{s}),\ 0 \text{ otherwise} \ , \tag{6.17}$$

which gives, after simplification:

$$\frac{\partial \texttt{sparse-top}_i}{\partial s_j}(K, \mathbf{s}) = \delta_{ij} - \frac{1}{|M_K(\mathbf{s})|} \text{ if } i, j \in M_K(\mathbf{s}),\ 0 \text{ otherwise} \ . \tag{6.18}$$

We remind that the derivatives of the `sparsemax` write:

$$\frac{\partial \texttt{sparsemax}_i}{\partial s_j}(\mathbf{s}) = \delta_{ij} - \frac{1}{|S(\mathbf{s})|} \text{ if } i, j \in S(\mathbf{s}),\ 0 \text{ otherwise} \ . \tag{6.19}$$

Equations (6.18) and (6.19) are quasi-identical. In the derivatives of `sparse-top`, $M_K(\mathbf{s})$ plays the role that $S(\mathbf{s})$ plays in the `sparsemax`.

Let us write the jacobians in matrix notations. Let us denote $\mathbf{p} = \texttt{softmax}(\mathbf{s})$. Then the jacobian of the `softmax` then reads:

$$J_{\texttt{softmax}}(\mathbf{s}) = Diag(\mathbf{p}) - \mathbf{p}\mathbf{p}^T \tag{6.20}$$

Let $\mathbf{m}$ be an indicator vector whose i-th entry is 1 if $i \in M(\mathbf{s})$ and 0 otherwise. Then the jacobian of $\texttt{sparse-top}_K$ reads:

$$J_{\texttt{sparse-top}}(K, \mathbf{s}) = Diag(\mathbf{m}) - \frac{\mathbf{m}\mathbf{m}^T}{|M(\mathbf{s})|} \tag{6.21}$$

Let $\mathbf{v}$ be an indicator vector whose i-th entry is 1 if $i \in S(\mathbf{v})$ and 0 otherwise. Then the jacobian of `sparsemax` reads:

$$J_{\texttt{sparsemax}}(\mathbf{v}) = Diag(\mathbf{v}) - \frac{\mathbf{v}\mathbf{v}^T}{|S(\mathbf{v})|} \tag{6.22}$$

6.2.5 Multi-label loss function with `sparse-top`

We remind that the gradient of the softmax loss reads:

$$\nabla_{\mathbf{s}} \ell_{\text{CE}}(\mathbf{s}, y) = -\delta_y + \texttt{softmax}(\mathbf{s}). \tag{6.23}$$

If you attempt to minimize ℓ_{CE} with SGD, the optimum is reached when $\nabla_{\mathbf{s}} \ell_{\text{CE}}(\mathbf{s}, y)$ is equal to 0, *i.e.*, when $\texttt{softmax}(\mathbf{s}) = \delta_y$, that is when the $\texttt{softmax}$ is equal to the target vector, with one at the grund truth and 0 elsewhere (which never happens since the softmax has full support). For multi-label classification, the target for an example $x \in \mathcal{X}$ is a set $Y \in 2^{[L]}$ that can contain several labels. Our intent is to design a loss that exactly predicts at optimum Y. To this end, we wish to design a loss $\ell_{\texttt{sparse-top}}$ so that:

$$\nabla_{\mathbf{s}} \ell_{\texttt{sparse-top}}(\mathbf{s}, Y) = -\sum_{k \in Y} \delta_k + \texttt{sparse-top}(|Y|, \mathbf{s}) \tag{6.24}$$

This way, the gradient of this loss is 0 when $\texttt{sparse-top}$ is equal to 1 on ground truth coordinates and 0 otherwise. By integrating, this leads to:

$$\ell_{\texttt{sparse-top}}(\mathbf{s}, Y) = -\sum_{k \in Y} s_k + \sum_{k \in H_{|Y|}(\mathbf{s})} \left(s_k - \frac{1}{2} \right) + \frac{1}{2} \sum_{k \in M_{|Y|}(\mathbf{s})} (s_k^2 - \lambda^2(\mathbf{s})) + \frac{|Y|}{2} \tag{6.25}$$

6.2.6 Properties

The following properties make $\ell_{\texttt{sparse-top}}$ a suitable loss for machine learning: it is a convex, differentiable loss.

Proposition 6.2.1.

- $\ell_{\texttt{sparse-top}}$ *is convex*
- $\forall \mathbf{s}, Y : \ell_{\texttt{sparse-top}}(\mathbf{s}, Y) \geq 0$
- $\ell_{\texttt{sparse-top}}(\mathbf{s}, Y) = 0 \Leftrightarrow \texttt{sparse-top}(|Y|, \mathbf{s}) = \sum_{k \in Y} \delta_k$

Proof.

- The hessian is given by:

$$\frac{\partial^2 \ell_{\texttt{sparse-top}}}{\partial s_i \partial s_j}(\mathbf{s}, Y) = \delta_{ij} - \frac{1}{|M_{|Y|}(\mathbf{s})|} \text{ if } i, j \in M_{|Y|}(\mathbf{s}),\ 0 \text{ otherwise.} \tag{6.26}$$

  It is positive semi-definite so $\ell_{\texttt{sparse-top}}$ is convex.

- For the second and last point, the minimum is attained when its gradient is zero *i.e.*, when $\texttt{sparse-top}(|Y|, \mathbf{s}) = \sum_{k \in Y} \delta_k$, in which case $M_{|Y|}(\mathbf{s}) = \emptyset$, $H_{|Y|}(\mathbf{s}) = Y$, and $\ell_{\texttt{sparse-top}}(\mathbf{s}, Y) = -\sum_{k \in Y} s_k + \sum_{k \in H_{|Y|}(\mathbf{s})} s_k = 0$

□

6.2.7 Open questions

One of the question with $\ell_{\texttt{sparse-top}}$ is what happens at prediction time. Indeed, during training, one has access to the labels Y and hence it is possible to project on $\Delta_{|Y|}^{L-1}$. However, at test time, we do not normally have access to the label and therefore we can not project on $\Delta_{|Y|}^{L-1}$. As a consequence, for a test example x, the best we can do is

to obtain the associated predicted logits $\mathbf{s} \in \mathbb{R}^L$ after inference. Notice that it is still possible to compute usual multi-label metrics with the logits so this is not a major problem. In addition, it would be possible to compute $\ell_{\texttt{sparse-top}}$ if one had access not to the full label Y, but merely to its cardinal $|Y|$. Therefore, if for some application and for some reason we knew the number of classes present in an input image x, we could compute the `sparse-top` of the logits and use this as a prediction. This would have the advantage to be sparse, which can be desired in some contexts. Finally, we believe that $\ell_{\texttt{sparse-top}}$ could be useful in other contexts where we need a vector on the top-K simplex, for instance in K-nearest neighbors. We are currently investing possible applications.

BIBLIOGRAPHY

[Aff+17] Antoine Affouard, Hervé Goëau, Pierre Bonnet, Jean-Christophe Lombardo, and Alexis Joly. "Pl@ntNet app in the era of deep learning". In: *ICLR - Workshop Track*. 2017 (cit. on pp. 3, 26, 44, 62).

[Ali+80] ARJH Ali, R Helgason, J Kennington, and H Lall. "Computational comparison among three multicommodity network flow algorithms". In: *Operations Research* 28.4 (1980), pp. 995–1000 (cit. on p. 89).

[ALS22] Md Manjurul Ahsan, Shahana Akter Luna, and Zahed Siddique. "Machine-learning-based disease diagnosis: A comprehensive review". In: *Healthcare*. Vol. 10. 3. MDPI. 2022, p. 541 (cit. on p. 16).

[Arn14] Barry C Arnold. "Pareto distribution". In: *Wiley StatsRef: Statistics Reference Online* (2014), pp. 1–10 (cit. on p. 2).

[Bat+20] Gopi Battineni, Getu Gamo Sagaro, Nalini Chinatalapudi, and Francesco Amenta. "Applications of machine learning predictive models in the chronic disease diagnosis". In: *Journal of personalized medicine* 10.2 (2020), p. 21 (cit. on p. 16).

[Ben+09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48 (cit. on p. 14).

[Ber+20] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. "Learning with differentiable perturbed optimizers". In: *NeurIPS*. 2020 (cit. on pp. 44, 45, 49, 50, 60, 61).

[BH81] Gabriel R Bitran and Arnoldo C Hax. "Disaggregation and resource allocation using convex knapsack problems with bounded variables". In: *Management Science* 27.4 (1981), pp. 431–441 (cit. on p. 89).

[BJM06] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. "Convexity, classification, and risk bounds". In: *J. Amer. Statist. Assoc.* 101.473 (2006), pp. 138–156 (cit. on p. 23).

[BLM13] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Second. Oxford University Press, 2013 (cit. on p. 61).

[Bru84] Peter Brucker. "An O (n) algorithm for quadratic knapsack problems". In: *Operations Research Letters* 3.3 (1984), pp. 163–166 (cit. on p. 89).

[BT12] Amir Beck and Marc Teboulle. "Smoothing and first order methods: A unified framework". In: *SIAM J. Optim.* 22.2 (2012), pp. 557–580 (cit. on p. 45).

[Bus+20] Aurelia Bustos, Antonio Pertusa, Jose-Maria Salinas, and Maria De La Iglesia-Vaya. "Padchest: A large chest x-ray image dataset with multi-label

annotated reports". In: *Medical image analysis* 66 (2020), p. 101797 (cit. on p. 86).

[BZK18] Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. "Smooth Loss Functions for Deep Top-k Classification". In: *ICLR*. 2018 (cit. on pp. 28, 32, 35, 41, 44, 45, 47–49, 52–56, 58, 62, 63, 70, 79).

[Cao+19] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. "Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss". In: *NeurIPS*. Vol. 32. 2019, pp. 1565–1576 (cit. on pp. 38, 44, 45, 47, 52, 57, 58).

[CAS16] Paul Covington, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations". In: *RecSys*. 2016, pp. 191–198 (cit. on pp. 44, 55).

[CB+04] Nicolo Cesa-Bianchi, Claudio Gentile, Andrea Tironi, and Luca Zaniboni. "Incremental algorithms for hierarchical classification". In: *Advances in neural information processing systems* 17 (2004) (cit. on p. 15).

[CDM16] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. "Learning with rejection". In: *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings 27*. Springer. 2016, pp. 67–82 (cit. on p. 14).

[CH94] Steven Cosares and Dorit S Hochbaum. "Strongly polynomial algorithms for the quadratic transportation problem with a fixed number of sources". In: *Mathematics of Operations Research* 19.1 (1994), pp. 94–111 (cit. on p. 89).

[Chz+21] Evgenii Chzhen, Christophe Denis, Mohamed Hebiri, and Titouan Lorieul. "Set-valued classification – overview via a unified framework". In: *arXiv preprint arXiv:2102.12318* (2021) (cit. on pp. 15, 26, 27, 58, 68, 69, 82).

[CM87] Paul H Calamai and Jorge J Moré. "Quasi-Newton updates with bounds". In: *SIAM journal on numerical analysis* 24.6 (1987), pp. 1434–1441 (cit. on p. 89).

[Col+20] Elijah Cole, Benjamin Deneu, Titouan Lorieul, Maximilien Servajean, Christophe Botella, Dan Morris, Nebojsa Jojic, Pierre Bonnet, and Alexis Joly. "The geolifeclef 2020 dataset". In: *arXiv preprint arXiv:2004.04192* (2020) (cit. on p. 55).

[Col+21] Elijah Cole, Oisin Mac Aodha, Titouan Lorieul, Pietro Perona, Dan Morris, and Nebojsa Jojic. "Multi-Label Learning from Single Positive Labels". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021 (cit. on pp. 71, 74, 75, 78).

[CS01] Koby Crammer and Yoram Singer. "On the algorithmic implementation of multiclass kernel-based vector machines". In: *J. Mach. Learn. Res.* 2.Dec (2001), pp. 265–292 (cit. on p. 48).

[Cui+17] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge J. Belongie. "Kernel Pooling for Convolutional Neural Networks". In: *CVPR*. 2017, pp. 3049–3058 (cit. on pp. 36, 38).

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20 (1995), pp. 273–297 (cit. on p. 22).

[DC00] Susan Dumais and Hao Chen. "Hierarchical classification of web content". In: *Proceedings of the 23rd annual international ACM SIGIR conference on*

Research and development in information retrieval. 2000, pp. 256–263 (cit. on p. 15).

[Dem+12] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. "On label dependence and loss minimization in multi-label classification". In: *Machine Learning* 88 (2012), pp. 5–45 (cit. on pp. 15, 71).

[DH17] Christophe Denis and Mohamed Hebiri. "Confidence Sets with Expected Sizes for Multiclass Classification". In: *J. Mach. Learn. Res.* 18 (2017), 102:1–102:28 (cit. on pp. 8, 17, 19, 28, 31, 68, 70).

[DKD09a] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural safety* 31.2 (2009), pp. 105–112 (cit. on p. 26).

[DKD09b] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural safety* 31.2 (2009), pp. 105–112 (cit. on p. 68).

[DKS04] Ofer Dekel, Joseph Keshet, and Yoram Singer. "Large margin hierarchical classification". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 27 (cit. on p. 15).

[Dos+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR*. OpenReview.net, 2021 (cit. on p. 32).

[DS69] Stella C Dafermos and Frederick T Sparrow. "The traffic assignment problem for a general network". In: *Journal of Research of the National Bureau of Standards B* 73.2 (1969), pp. 91–118 (cit. on p. 89).

[Eve+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88 (2010), pp. 303–338 (cit. on p. 15).

[Fav+20] Carole Faviez, Xiaoyi Chen, Nicolas Garcelon, Antoine Neuraz, Bertrand Knebelmann, Rémi Salomon, Stanislas Lyonnet, Sophie Saunier, and Anita Burgun. "Diagnosis support systems for rare diseases: a scoping review". In: *Orphanet Journal of Rare Diseases* 15.1 (2020), pp. 1–16 (cit. on p. 80).

[FB+21] Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. "The limits of distribution-free conditional predictive inference". In: *Information and Inference: A Journal of the IMA* 10.2 (2021), pp. 455–482 (cit. on p. 17).

[FHT00] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". In: *The annals of statistics* 28.2 (2000), pp. 337–407 (cit. on p. 22).

[FS97] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139 (cit. on p. 22).

[FZM17] Jianlong Fu, Heliang Zheng, and Tao Mei. "Look Closer to See Better: Recurrent Attention Convolutional Neural Network for Fine-Grained Image Recognition". In: *CVPR*. 2017, pp. 4476–4484 (cit. on pp. 36, 38).

[FZV23] Matteo Fontana, Gianluca Zeni, and Simone Vantini. "Conformal prediction: a unified review of theory and new challenges". In: *Bernoulli* 29.1 (2023), pp. 1–23 (cit. on pp. 16, 68, 70).

[Gar+21] Camille Garcin, Alexis Joly, Pierre Bonnet, Antoine Affouard, Jean-Christophe Lombardo, Mathias Chouet, Maximilien Servajean, Titouan Lorieul, and Joseph Salmon. "Pl@ntNet-300K: a plant image dataset with high label ambiguity and a long-tailed distribution". In: *NeurIPS Datasets and Benchmarks 2021*. 2021 (cit. on pp. 11, 44, 52, 56, 62, 68, 69, 71, 72, 79).

[Gar+22] Camille Garcin, Maximilien Servajean, Alexis Joly, and Joseph Salmon. "Stochastic smoothing of the top-K calibrated hinge loss for deep imbalanced classification". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 7208–7222 (cit. on pp. 11, 70, 78, 79).

[Gar+23] Camille Garcin, Maximilien Servajean, Alexis Joly, and Joseph Salmon. *A two-head loss function for deep Average-K classification*. Tech. rep. 2023 (cit. on p. 11).

[Gas71] Joseph L Gastwirth. "A general definition of the Lorenz curve". In: *Econometrica: Journal of the Econometric Society* (1971), pp. 1037–1039 (cit. on p. 29).

[Ge+16] ZongYuan Ge, Alex Bewley, Christopher McCool, Peter I. Corke, Ben Upcroft, and Conrad Sanderson. "Fine-grained classification via mixture of deep convolutional neural networks". In: *CVPR*. 2016, pp. 1–6 (cit. on p. 36).

[Geb+17] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. "Fine-grained car detection for visual census estimation". In: *AAAI*. Vol. 31. 1. 2017 (cit. on p. 36).

[Gor87] Allan D Gordon. "A review of hierarchical classification". In: *Journal of the Royal Statistical Society: Series A (General)* 150.2 (1987), pp. 119–137 (cit. on p. 14).

[Ha97] Thien M Ha. "The optimum class-selective rejection rule". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6 (1997), pp. 608–615 (cit. on p. 19).

[Har+23] Adam G Hart, Hayley Bosley, Chloe Hooper, Jessica Perry, Joel Sellors-Moore, Oliver Moore, and Anne E Goodenough. "Assessing the accuracy of free automated plant identification applications". In: *People and Nature* (2023) (cit. on p. 68).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *CVPR*. 2016, pp. 770–778 (cit. on pp. 32, 56, 57, 79).

[HH95] Dorit S Hochbaum and Sung-Pil Hong. "About strongly polynomial time algorithms for quadratic optimization over submodular constraints". In: *Mathematical programming* 69.1-3 (1995), pp. 269–309 (cit. on p. 89).

[HKL80a] R Helgason, J Kennington, and H Lall. "A polynomially bounded algorithm for a singly constrained quadratic program". In: *Mathematical Programming* 18 (1980), pp. 338–343 (cit. on p. 89).

[HKL80b] Richard V. Helgason, Jeffery L. Kennington, and H. S. Lall. "A polynomially bounded algorithm for a singly constrained quadratic program". In: *Math. Program.* 18.1 (1980), pp. 338–343 (cit. on p. 60).

[Hor+18] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. "The INaturalist Species Classification and Detection Dataset". In: *CVPR*. 2018, pp. 8769–8778 (cit. on pp. 36, 44, 52, 68).

[How+19] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. "Searching for MobileNetV3". In: *ICCV*. 2019, pp. 1314–1324 (cit. on p. 32).

[Hua+17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CVPR*. 2017, pp. 2261–2269 (cit. on pp. 32, 53, 56, 79).

[HW06] Radu Herbei and Marten H Wegkamp. "Classification with reject option". In: *The Canadian Journal of Statistics/La Revue Canadienne de Statistique* (2006), pp. 709–721 (cit. on p. 14).

[HWC19] Xiangyu He, Peisong Wang, and Jian Cheng. "K-Nearest Neighbors Hashing". In: *CVPR*. 2019, pp. 2839–2848 (cit. on p. 44).

[Ian+16] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". In: *CoRR* abs/1602.07360 (2016). arXiv: 1602.07360 (cit. on p. 32).

[IMV19] Arya Iranmehr, Hamed Masnadi-Shirazi, and Nuno Vasconcelos. "Cost-sensitive support vector machines". In: *Neurocomputing* 343 (2019), pp. 50–64 (cit. on pp. 45, 52).

[Kha17] Faiza Khan Khattak. "Toward a Robust and Universal Crowd Labeling Framework". PhD thesis. Columbia University, 2017 (cit. on p. 71).

[Khu+23] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. "Natural language processing: State of the art, current trends and challenges". In: *Multimedia tools and applications* 82.3 (2023), pp. 3713–3744 (cit. on p. 2).

[Kiw07] Krzysztof C Kiwiel. "On linear-time algorithms for the continuous quadratic knapsack problem". In: *Journal of Optimization Theory and Applications* 134 (2007), pp. 549–554 (cit. on p. 89).

[Kiw08] KC Kiwiel. "Variable fixing algorithms for the continuous quadratic knapsack problem". In: *Journal of Optimization Theory and Applications* 136 (2008), pp. 445–458 (cit. on p. 89).

[Kri09] Alex Krizhevsky. "Learning multiple layers of features from tiny images". MA thesis. University of Toronto, 2009 (cit. on pp. 2, 38, 44, 55, 71, 72, 79).

[KS97] Daphne Koller and Mehran Sahami. "Hierarchically classifying documents using very few words". In: *ICML*. Vol. 97. 1997, pp. 170–178 (cit. on p. 15).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NeurIPS*. 2012, pp. 1106–1114 (cit. on p. 32).

[LeC+89] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Wayne Hubbard, and Lawrence Jackel. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems* 2 (1989) (cit. on p. 2).

[LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on pp. 2, 44).

[Lei14] Jing Lei. "Classification with confidence". In: *Biometrika* 101.4 (2014), pp. 755–769 (cit. on p. 17).

[LHS15] Maksim Lapin, Matthias Hein, and Bernt Schiele. "Top-k multiclass SVM". In: *NeurIPS*. 2015, pp. 325–333 (cit. on pp. 7, 17, 18, 27, 28, 32, 44, 45, 47, 48, 52, 68, 70).

[LHS16] Maksim Lapin, Matthias Hein, and Bernt Schiele. "Loss functions for top-k error: Analysis and insights". In: *CVPR*. 2016, pp. 1468–1477 (cit. on pp. 17, 28, 31, 32, 45).

[LHS17] Maksim Lapin, Matthias Hein, and Bernt Schiele. "Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.7 (2017), pp. 1533–1554 (cit. on pp. 17, 27, 28, 32, 45).

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755 (cit. on p. 15).

[Lin+17] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. "Focal Loss for Dense Object Detection". In: *ICCV*. 2017, pp. 2999–3007 (cit. on pp. 45, 47).

[Lit+20] Damon P Little, Melissa Tulig, Kiat Chuan Tan, Yulong Liu, Serge Belongie, Christine Kaeser-Chen, Fabián A Michelangeli, Kiran Panesar, RV Guha, and Barbara A Ambrose. "An algorithm competition for automatic species identification from herbarium specimens". In: *Applications in plant sciences* 8.6 (2020), e11365 (cit. on p. 72).

[Liu+19] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. "Large-Scale Long-Tailed Recognition in an Open World". In: *CVPR*. 2019, pp. 2537–2546 (cit. on p. 58).

[Liu+21] Weiwei Liu, Haobo Wang, Xiaobo Shen, and Ivor W Tsang. "The emerging trends of multi-label learning". In: *IEEE transactions on pattern analysis and machine intelligence* 44.11 (2021), pp. 7955–7974 (cit. on pp. 15, 71).

[LJS21] Titouan Lorieul, Alexis Joly, and Dennis Shasha. "Classification Under Ambiguity: When Is Average-K Better Than Top-K?" In: *arXiv preprint arXiv:2112.08851* (2021) (cit. on pp. 21, 24).

[Lor20] Titouan Lorieul. "Uncertainty in predictions of deep learning models for fine-grained classification". PhD thesis. Université de Montpellier, Dec. 2020 (cit. on pp. 23, 35, 68, 70, 73, 76, 80, 81).

[Lor+22] Titouan Lorieul, Elijah Cole, Benjamin Deneu, Maximilien Servajean, Pierre Bonnet, and Alexis Joly. "Overview of GeoLifeCLEF 2022: Predict-

ing species presence from multi-modal remote sensing, bioclimatic and pedologic data". In: *CLEF 2022-Conference and Labs of the Evaluation Forum*. Vol. 3180. 155. 2022, pp. 1940–1956 (cit. on p. 80).

[LRM15] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear CNN Models for Fine-Grained Visual Recognition". In: *ICCV*. IEEE Computer Society, 2015, pp. 1449–1457 (cit. on pp. 36, 38).

[LST03] Yaoyong Li and John Shawe-Taylor. "The SVM with uneven margins and Chinese document categorization". In: *PACLIC*. 2003, pp. 216–227 (cit. on pp. 45, 52).

[LV04] Gábor Lugosi and Nicolas Vayatis. "On the Bayes-risk consistency of regularized boosting methods". In: *The Annals of statistics* 32.1 (2004), pp. 30–55 (cit. on p. 23).

[LW14] Jing Lei and Larry Wasserman. "Distribution-free prediction bands for non-parametric regression". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 76.1 (2014), pp. 71–96 (cit. on p. 17).

[MA16] Andre Martins and Ramon Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification". In: *International conference on machine learning*. PMLR. 2016, pp. 1614–1623 (cit. on p. 88).

[Mac+03] N Maculan, C Prata Santiago, EM Macambira, and MHC Jardim. "An O (n) algorithm for projecting a vector on the intersection of a hyperplane and a box in R n". In: *Journal of optimization theory and applications* 117 (2003), pp. 553–574 (cit. on p. 89).

[Mac+21] Andre Machado, Rodrigo Veras, Kelson Aires, and Laurindo de Sousa Britto Neto. "A systematic review on product recognition for aiding visually impaired people". In: *IEEE Latin America Transactions* 19.4 (2021), pp. 592–603 (cit. on p. 68).

[Maj+13] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. "Fine-grained visual classification of aircraft". In: *arXiv preprint arXiv:1306.5151* (2013) (cit. on pp. 26, 36).

[Mic86] Christian Michelot. "A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$". In: *Journal of Optimization Theory and Applications* 50 (1986), pp. 195–200 (cit. on p. 89).

[MPJ89] Nelson Maculan and Geraldo Galdino de Paula Jr. "A linear-time median-finding algorithm for projecting a vector on the simplex of Rn". In: *Operations research letters* 8.4 (1989), pp. 219–222 (cit. on p. 89).

[Nas+19] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. "Speech recognition using deep neural networks: A systematic review". In: *IEEE access* 7 (2019), pp. 19143–19165 (cit. on p. 2).

[Nes05] Yu Nesterov. "Smooth minimization of non-smooth functions". In: *Math. Program.* 103.1 (2005), pp. 127–152 (cit. on p. 45).

[NZ08] Maria-Elena Nilsback and Andrew Zisserman. "Automated flower classification over a large number of classes". In: *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE. 2008, pp. 722–729 (cit. on pp. 26, 36).

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *NeurIPS*. 2019, pp. 8026–8037 (cit. on p. 53).

[Pit+21] Nigel CA Pitman, Tomomi Suwa, Carmen Ulloa Ulloa, James Miller, James Solomon, Juliana Philipp, Corine F Vriesendorp, Abigail Derby Lewis, Sinem Perk, Pierre Bonnet, et al. "Identifying gaps in the photographic record of the vascular plant flora of the Americas". In: *Nature plants* 7.8 (2021), pp. 1010–1014 (cit. on p. 4).

[PK90] Panos M Pardalos and Naina Kovoor. "An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds". In: *Mathematical Programming* 46 (1990), pp. 321–328 (cit. on p. 89).

[Qia+15] Qi Qian, Rong Jin, Shenghuo Zhu, and Yuanqing Lin. "Fine-grained visual categorization via multi-stage metric learning". In: *CVPR*. 2015, pp. 3716–3724 (cit. on p. 36).

[Ree01] William J Reed. "The Pareto, Zipf and other power laws". In: *Economics letters* 74.1 (2001), pp. 15–19 (cit. on p. 44).

[RJL92] AG Robinson, N Jiang, and CS Lerme. "On the continuous quadratic knapsack problem". In: *Mathematical programming* 55 (1992), pp. 99–108 (cit. on p. 89).

[RMN09] Rajat Raina, Anand Madhavan, and Andrew Y Ng. "Large-scale deep unsupervised learning using graphics processors". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 873–880 (cit. on p. 2).

[Rob+19] Tim Robertson, Serge Belongie, Adam Hartwig, Christine Kaeser-Chen, Chenyang Zhang, Kiat Chuan Tan, Yulong Liu, Denis Brulé, Cédric Deltheil, Scott Loarie, et al. "Training machines to identify species using GBIF-mediated datasets". In: *Biodiversity Information Science and Standards* (2019) (cit. on p. 68).

[RSC20] Yaniv Romano, Matteo Sesia, and Emmanuel Candes. "Classification with valid and adaptive coverage". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3581–3591 (cit. on p. 17).

[Rud16a] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 32).

[Rud16b] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 71).

[Rus+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *Int. J. Comput. Vision* 115.3 (2015), pp. 211–252 (cit. on pp. 2, 26, 36, 38, 44, 55–57, 68, 71, 72, 79).

[San+18] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *CVPR*. 2018, pp. 4510–4520 (cit. on p. 32).

[Sco12] Clayton Scott. "Calibrated asymmetric surrogate losses". In: *Electron. J. Stat.* 6 (2012), pp. 958–992 (cit. on pp. 45, 52).

[SF11a] Carlos Silla and Alex Alves Freitas. "A survey of hierarchical classification across different application domains". In: *Data Min. Knowl. Discov.* 22.1-2 (2011), pp. 31–72 (cit. on p. 38).

[SF11b] Carlos N Silla and Alex A Freitas. "A survey of hierarchical classification across different application domains". In: *Data mining and knowledge discovery* 22 (2011), pp. 31–72 (cit. on p. 14).

[Sla+16] Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic. "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification". In: *Comput. Intell. Neurosci.* 2016 (2016), 3289801:1–3289801:11 (cit. on p. 36).

[SLW19] Mauricio Sadinle, Jing Lei, and Larry Wasserman. "Least ambiguous set-valued classifiers with bounded error levels". In: *Journal of the American Statistical Association* 114.525 (2019), pp. 223–234 (cit. on p. 17).

[SSZ13] Shai Shalev-Shwartz and Tong Zhang. "Stochastic dual coordinate ascent methods for regularized loss minimization". In: *J. Mach. Learn. Res.* 14.Feb (2013), pp. 567–599 (cit. on pp. 45, 70).

[SV08] Glenn Shafer and Vladimir Vovk. "A Tutorial on Conformal Prediction." In: *Journal of Machine Learning Research* 9.3 (2008) (cit. on pp. 16, 70).

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *ICLR*. 2015 (cit. on p. 32).

[Sze+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *CVPR*. 2016, pp. 2818–2826 (cit. on p. 32).

[Sze+17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *AAAI*. AAAI Press, 2017, pp. 4278–4284 (cit. on p. 32).

[TB07] Ambuj Tewari and Peter L Bartlett. "On the Consistency of Multiclass Classification Methods." In: *Journal of Machine Learning Research* 8.5 (2007) (cit. on p. 23).

[TL19] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *ICML*. Vol. 97. 2019, pp. 6105–6114 (cit. on p. 32).

[UBG09] Nicolas Usunier, David Buffoni, and Patrick Gallinari. "Ranking with ordered weighted pairwise classification". In: *ICML*. Vol. 382. 2009, pp. 1057–1064 (cit. on p. 44).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 2).

[VGS05] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Vol. 29. Springer, 2005 (cit. on pp. 16, 70).

[VH+15] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. "Building a bird recognition app and large scale dataset with citizen scientists: The fine print

in fine-grained dataset collection". In: *CVPR*. 2015, pp. 595–604 (cit. on p. 44).

[Vou+18] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018 (2018) (cit. on p. 2).

[Wan+18] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. "Additive Margin Softmax for Face Verification". In: *IEEE Trans. Signal Process. Lett.* 25.7 (2018), pp. 926–930 (cit. on p. 57).

[Wan+21] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella X. Yu. "Long-tailed Recognition by Routing Diverse Distribution-Aware Experts". In: *ICLR*. 2021 (cit. on pp. 44, 45).

[Wan+22] Jingdong Wang, Zhuowen Tu, Jianlong Fu, Nicu Sebe, and Serge Belongie. "Guest Editorial: Introduction to the Special Section on Fine-Grained Visual Categorization". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.02 (2022), pp. 560–562 (cit. on p. 44).

[WCZ21] Xin Wang, Yudong Chen, and Wenwu Zhu. "A survey on curriculum learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9 (2021), pp. 4555–4576 (cit. on p. 14).

[Wel+10] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010 (cit. on p. 36).

[WWC19] Xiu-Shen Wei, Jianxin Wu, and Quan Cui. "Deep learning for fine-grained image analysis: A survey". In: *arXiv preprint arXiv:1907.03069* (2019) (cit. on p. 2).

[XE19] Sang Michael Xie and Stefano Ermon. "Reparameterizable subset sampling via continuous relaxations". In: *IJCAI*. ijcai.org, 2019, pp. 3919–3925 (cit. on p. 45).

[Xie+20] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. "Differentiable Top-k with Optimal Transport". In: *NeurIPS*. 2020 (cit. on p. 45).

[Yan+15] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. "A large-scale car dataset for fine-grained categorization and verification". In: *CVPR*. 2015, pp. 3973–3981 (cit. on pp. 26, 36).

[YK20] Forest Yang and Sanmi Koyejo. "On the consistency of top-k surrogate losses". In: *ICML*. Vol. 119. 2020, pp. 10727–10735 (cit. on pp. 24, 45, 47–50, 52, 58, 59, 70).

[Yua+21] Jiangbo Yuan, An-Ti Chiang, Wen Tang, and Antonio Haro. "eproduct: A million-scale visual search benchmark to address product recognition challenges". In: *arXiv preprint arXiv:2107.05856* (2021) (cit. on p. 68).

[Zha04] Tong Zhang. "Statistical behavior and consistency of classification methods based on convex risk minimization". In: *The Annals of Statistics* 32.1 (2004), pp. 56–85 (cit. on p. 23).

[Zha+18] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". In: *CVPR*. 2018, pp. 6848–6856 (cit. on p. 32).

[Zho+20] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. "BBN: Bilateral-Branch Network With Cumulative Learning for Long-Tailed Visual Recognition". In: *CVPR*. 2020, pp. 9716–9725 (cit. on pp. 38, 45).

[ZK16] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *BMVC*. 2016 (cit. on p. 32).

[ZZ06] Min-Ling Zhang and Zhi-Hua Zhou. "Multilabel neural networks with applications to functional genomics and text categorization". In: *IEEE transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1338–1351 (cit. on p. 86).

[ZZ14] Min-Ling Zhang and Zhi-Hua Zhou. "A Review on Multi-Label Learning Algorithms". In: *IEEE Trans. Knowl. Data Eng.* 26.8 (2014), pp. 1819–1837 (cit. on pp. 15, 27).