A two-head loss function for deep Average-K classification

Camille Garcin, Maximilien Servajean, Joseph Salmon, Alexis Joly

A two-head loss function for deep Average-K classification

Camille Garcin[1,2], Maximilien Servajean[3,4], Alexis Joly[2,3], Joseph Salmon[1,5]
[1]IMAG, Univ Montpellier, CNRS, Montpellier, France
[2]Inria, LIRMM, Univ Montpellier, CNRS, Montpellier, France
[3]LIRMM, Univ Montpellier, CNRS, Montpellier, France
[4]AMIS, Paul Valery University, Montpellier, France
[5]Institut Universitaire de France (IUF)
camille.garcin@inria.fr, maximilien.servajean@lirmm.fr
alexis.joly@inria.fr, joseph.salmon@umontpellier.fr

Abstract

Average-K classification is an alternative to top-K classification in which the number of labels returned varies with the ambiguity of the input image but must average to K over all the samples. 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.

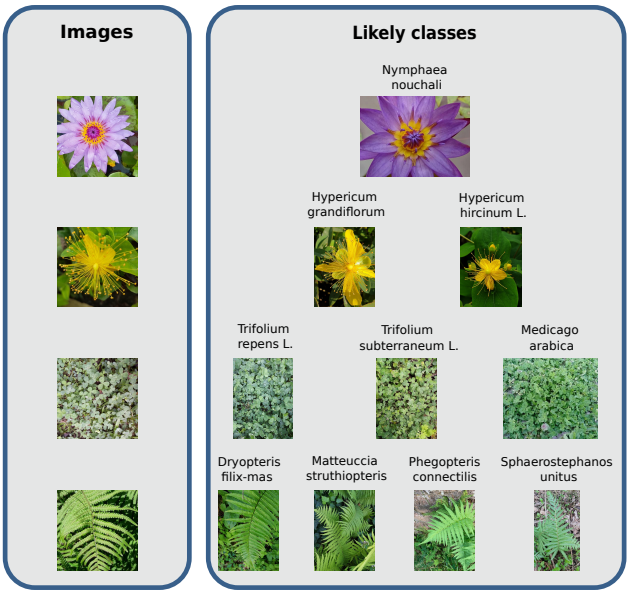


Figure 1: Variety of ambiguity in Pl@ntNet-300K dataset [9]. The images on the left are from the dataset while the ones on the right represent the likely corresponding classes. The first row corresponds to the case where there is almost no uncertainty, the second row to the case where two classes are viable candidates for the image, etc.

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 [13, 9, 30, 23]. 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 [6]. This is problematic for systems that aim to provide their users with a correct answer from an image [11, 22].

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 [17]. For instance, top-K accuracy is the official

ImageNet [25] 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 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 [2] and strategies [8], some more flexible than others. In this paper, we focus on average-K classifiers [5] 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 [20], no guarantee exists in the finite sample case.

In this paper, 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 cross-entropy training and other methods.

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 $y \in [L] \coloneqq \{1, \dots, L\}$, where L is the number of classes. Traditional multi-class classifiers are functions $f : \mathcal{X} \to [L]$ 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 $g : \mathcal{X} \to 2^{[L]}$ that map an image to a set of labels [2], where $2^{[L]}$ is the power set of $[L]$. Our objective is then to build a classifier g with minimal risk $\mathcal{R}(g) \coloneqq \mathbb{P}(Y \notin g(X))$.

A trivial solution would be to take $g(x) = [L]$ 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 [2] where a unified framework encompassing several possible constraints is proposed). In this paper, we focus on the average set size constraint that controls the average size of the returned set:

$$\mathbb{E}_X[|g(X)|] \leq K \ \ , (\mathcal{C}) \tag{1}$$

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

It has been shown [5] that the optimal classifier g^* minimizing $\mathcal{R}$ while satisfying $\mathcal{C}$ is:

$$g^*(x) = \{j \in [L], \mathbb{P}(Y = j | X = x) \geq \lambda\} \ , \tag{2}$$

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 [20], there is no guarantee in the finite sample regime. In this paper, we propose an alternative to cross-entropy to optimize average-K accuracy by formulating the problem as a multi-label problem.

3. Related work

Several works have studied set-valued classification. The most studied case is top-K classification. In [17], the authors propose a top-K hinge loss and optimize it with Stochastic Dual Coordinate Ascent [27]. In [29], the authors propose a slight variation of the top-K hinge loss with stronger theoretical guarantees. In [1] and [10], the authors propose to smooth top-K hinge losses to make them suitable for deep learning.

Average-K classification is less studied. In [5], 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 [20], 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 [28, 26, 8]. 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 $\hat{f} : \mathcal{X} \to [L]$, 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 [9]).

Set-valued classification can also be connected to multi-label classification [4, 19]. 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 [3] 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.

4. Set-valued classification as multi-label

4.1. Preliminaries

In most multi-class datasets [9, 16, 25], 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 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 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 [15].

In this paper, 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.

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 paper, we use mini-batch gradient descent [24] 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 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:

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

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:

$$\ell_{\text{CE}}(\mathbf{z}_i, y_i) = -\log(\varsigma_{y_i}(\mathbf{z}_i)) . \tag{3}$$

The partial derivatives read:

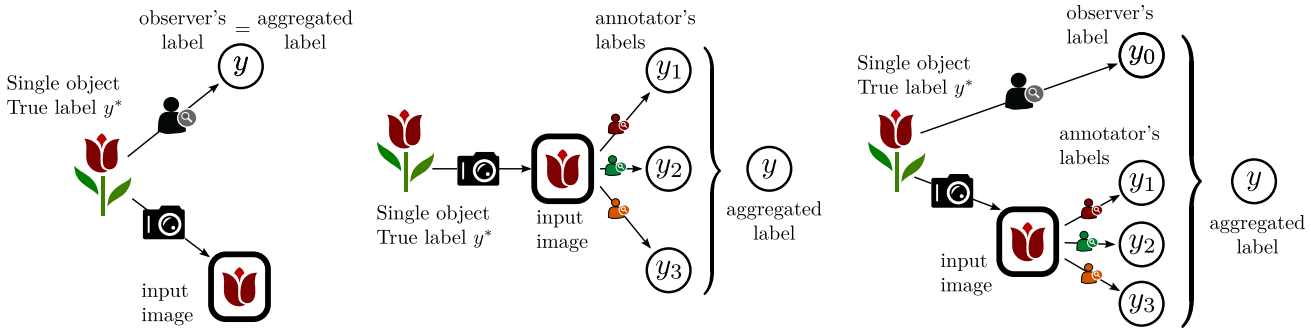
$$\frac{\partial \ell_{\text{CE}}}{\partial z_j}(\mathbf{z}_i, y_i) = \begin{cases} \varsigma_{y_i}(\mathbf{z}_i)(1 - \varsigma_{y_i}(\mathbf{z}_i)), & \text{if } j = y_i \\ -\varsigma_{y_i}(\mathbf{z}_i) \cdot \varsigma_j(\mathbf{z}_i), & \text{o.w.} \end{cases} , \tag{4}$$

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 1 to 3, in the infinite limit case, ℓ_{CE} has theoretical grounds [20]. 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:

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



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 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.,* [18] b) by aggregating the labels of annotators having access only to a partial representation of the object (the input image), *e.g.,* [16, 25] c) with a combination of both previous cases *e.g.,* [9]. In all cases, annotations errors can occur resulting in a final label y different than the true class y^*.

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.

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} [3], essentially the binary cross entropy-loss with a single positive label:

$$\ell_{\text{AN}}(\mathbf{z}_i, y_i) = -\sum_{j=1}^{L} \Big[\mathbb{1}_{[j=y_i]} \log(\sigma(z_{ij})) + \tfrac{1}{L-1} \mathbb{1}_{[j\neq y_i]} \log(1-\sigma(z_{ij})) \Big] \ , \quad (5)$$

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

4.5. Expected positive regularization

The problem with ℓ_{AN} is that the second term of Equation (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 [3]:

$$\ell^{+}_{\text{BCE}}(\mathbf{z}_i, y_i) = -\sum_{j=1}^{L} \mathbb{1}_{[j=y_i]} \log(\sigma(z_{ij})) \ . \quad (6)$$

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 [3] 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:

$$\hat{K}(Z^B) = \frac{1}{|B|} \sum_{i\in B} \sum_{j=1}^{L} \sigma(z_{ij}) \ . \quad (7)$$

The Expected positive regularization loss [3] ℓ_{EPR} then reads:

$$\ell_{\text{EPR}}(Z^B; Y^B) = \tfrac{-1}{|B|} \sum_{i\in B} \log(\sigma(z_{i,y_i})) + \beta(\hat{K}(Z^B) - K)^2, \quad (8)$$

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

Although the idea behind ℓ_{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.

4.6. Online estimation of labels

In [3], the authors introduce a loss ℓ_{ROLE} that builds on ℓ_{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 (8). The role of the predicted and estimated labels is then reversed. For more details, we refer the reader to [3].

Although ℓ_{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. Proposed method

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:

$$\begin{aligned}\ell(Z^B, Y^B, \mathcal{S}_c^B) = &- \frac{1}{|B|} \sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j=y_i]} \log(\sigma(z_{ij})) \\ &- \alpha \frac{\sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \in \mathcal{S}_c(x_i)]} \log(\sigma(z_{ij}))}{\sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \in \mathcal{S}_c(x_i)]}} \\ &- \alpha \frac{\sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \notin \mathcal{S}(x_i)]} \log(1 - \sigma(z_{ij}))}{\sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}_{[j \notin \mathcal{S}(x_i)]}} \quad ,\end{aligned} \tag{9}$$

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 $\mathcal{S}_c(x_i)$?

5.2. 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 (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 [20]. More formally, we optimize the SCCP head with:

$$\ell_{\text{CE}}(Z'^B, Y^B) = -\frac{1}{|B|} \sum_{i \in B} \log(\varsigma_{y_i}(\mathbf{z}'_i)) \quad , \tag{10}$$

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:

$$\Sigma_{ij}^{-\infty} = \begin{cases} \varsigma_j(\mathbf{z}'_i), & \text{if } y_i \neq j \\ -\infty, & \text{otherwise} \end{cases} \quad . \tag{11}$$

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:

$$\mathcal{S}_c(x_i) = \{j, \Sigma_{ij}^{-\infty} \text{ is in the top-}(K-1)|B| \text{ values of } \Sigma^{-\infty}\} \quad . \tag{12}$$

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

$$\begin{aligned}\frac{1}{|B|} \sum_{i \in B} |\mathcal{S}(x_i)| &= \frac{1}{|B|} \sum_{i \in B} |\mathcal{S}_c(x_i) \cup \{y_i\}| \\ &= \frac{(K-1)|B| + |B|}{|B|} \\ &= K\end{aligned} \tag{13}$$

An illustrative schema of the method is available in Figure 3. We can now plug the estimated candidate sets $\mathcal{S}_c^B$ into Equation (9):

$$\begin{aligned}\ell_{\text{BCE}}(Z^B, Y^B, \mathcal{S}_c^B) = &- \tfrac{1}{|B|} \sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}[j = y_i] \log(\sigma(z_{ij})) \\ &- \tfrac{\alpha}{(K-1)|B|} \sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}[j \in \mathcal{S}_c(x_i)] \log(\sigma(z_{ij})) \\ &- \tfrac{\alpha}{(L-K)|B|} \sum_{\substack{i \in B \\ j \in [L]}} \mathbb{1}[j \notin \mathcal{S}(x_i)] \log(1 - \sigma(z_{ij})) \quad ,\end{aligned} \tag{14}$$

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

$$\ell_{\text{AVG-K}}(Z'^B, Z^B, Y^B) = \ell_{\text{CE}}(Z'^B, Y^B) + \ell_{\text{BCE}}(Z^B, Y^B, \mathcal{S}_c^B) \tag{15}$$

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

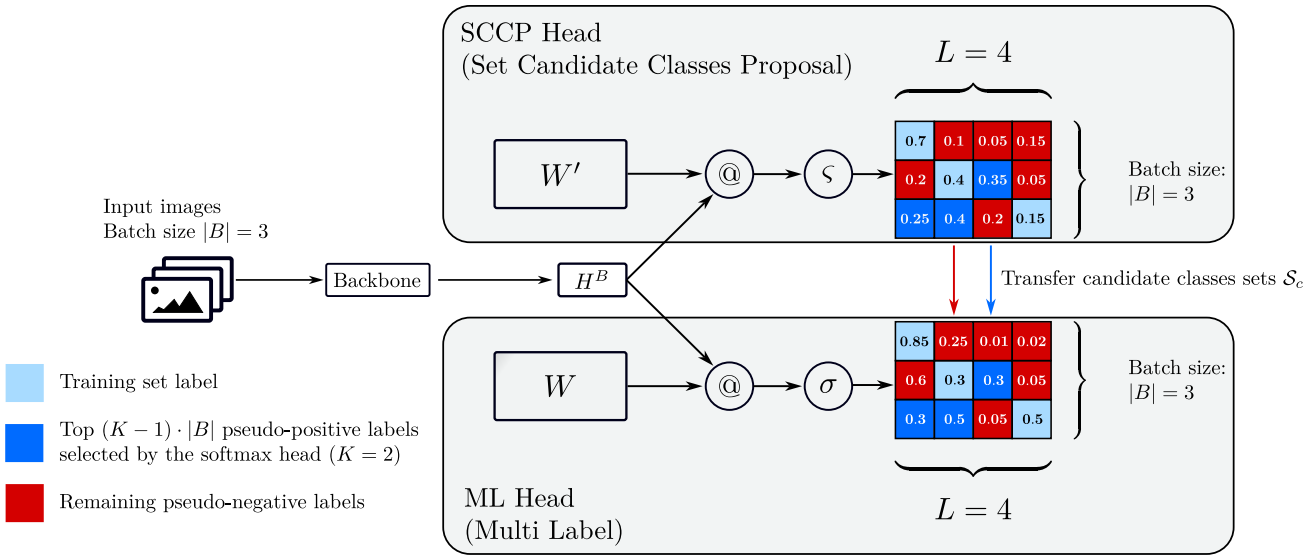


Figure 3: The Set Candidate Classes Proposal -SCCP- head (on top) is responsible for determining, for each example of the batch, which classes to include in the target label set. The light blue cells correspond to the training set labels. Only one label is assigned per example/row. The dark blue cells 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.

5.3. 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.

5.4. 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 [3], which is prohibitive for large datasets. Besides, instead of a constraint on the average value of the predictions [3], 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 : $n_{val} := |\mathcal{N}_{val}|$, $X_{val} = \{x_i\}_{i\in\mathcal{N}_{val}}$, 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

  $P \leftarrow \text{CONCAT}(P, p_i)$ // row axis concat.

/* At this point P is a $n_{val} \times L$ matrix */

$P \leftarrow \text{FLATTEN}(P)$ // turn P into a vector

$P \leftarrow \text{SORT}(P)$ // sort in decreasing order

$\lambda_{val} \leftarrow \frac{1}{2}(P_{[Kn_{val}]} + P_{[Kn_{val}+1]})$

return λ_{val}

ℓ_{CE}	$\ell_{\text{AVG-K}}$	ℓ_{ROLE}	ℓ_{AN}	ℓ_{EPR}	$\ell_{\text{TOP-K}}$
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 1: CIFAR-100 test average-5 accuracy, (DenseNet 40-40)

6. Experiments

6.1. Metrics

We compare $\ell_{\text{AVG-K}}$ with ℓ_{CE}, ℓ_{AN}, ℓ_{EPR} and ℓ_{ROLE} on two datasets with different degrees of ambiguity. We also include the balanced top-K loss $\ell_{\text{TOP-K}}$ from [10]. 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 average-K accuracy, computed as follows:

$$\textit{val avg-K accuracy} = \frac{1}{|\mathcal{N}_{val}|} \sum_{i \in \mathcal{N}_{val}} \mathbb{1}[\varsigma_{y_i}(z_i) \geq \lambda_{val}] \ , \tag{16}$$

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}:

$$\textit{test avg-K accuracy} = \frac{1}{|\mathcal{N}_{test}|} \sum_{i \in \mathcal{N}_{test}} \mathbb{1}[\varsigma_{y_i}(z_i) \geq \lambda_{val}] \ . \tag{17}$$

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.

6.2. CIFAR100

Training: We first experiment our method on CIFAR-100 [16], 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 [14] for 300 epochs with SGD and a Nesterov momentum of 0.9, following [10, 1]. 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 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 1.) This relatively low ambiguity explains the good performances of ℓ_{CE} and ℓ_{AN}. We find that ℓ_{EPR} and ℓ_{ROLE} lag behind, while $\ell_{\text{AVG-K}}$ based on the proposal of candidate sets benefits from a performance gain over all the other methods.

6.3. Pl@ntNet-300K

Description: Pl@ntNet-300K [9] 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 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.

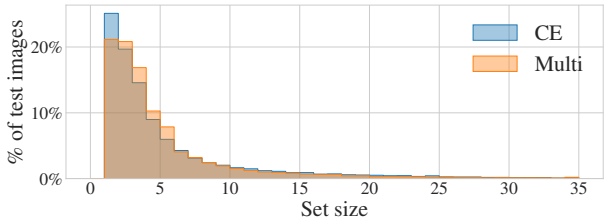
Training: We finetune a ResNet-18 [12] pre-trained on ImageNet [25] for 30 epochs. We use SGD and a Nesterov momentum of 0.9, with a batch size of 32 and an initial learning rate of 0.002, divided by ten at epoch 25. The weight decay is fixed at 0.0001. All methods are trained for $K \in \{2, 3, 5, 10\}$ 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 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 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 [7] or for the automatic prediction of likely species at a particular location [21].

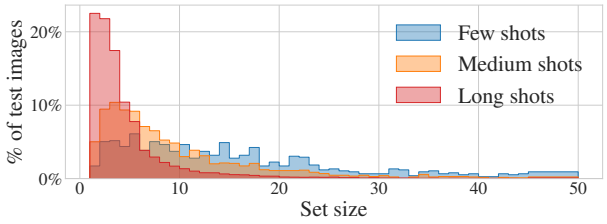
From Table 2 we see that the most naive methods ℓ_{AN} and ℓ_{EPR} perform poorly compared to the other losses. In particular, ℓ_{AN} which gave decent performances on CIFAR-

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

Table 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).



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

Figure 4: Histograms of set sizes.

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 $\ell_{\text{TOP-K}}$ 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 [20].

Distribution of set sizes: Figure 4a shows the repartition of set sizes for Pl@ntNet-300K test images, for a model trained with $\ell_{\text{AVG-5}}$ 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 4b shows the distribution of set size for few-shot, medium-shot and long-shot classes for a model trained with $\ell_{\text{AVG-5}}$ 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.

7. Limitations and future work

While $\ell_{\text{AVG-K}}$ 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 $\ell_{\text{AVG-K}}$ is average-K calibrated [20] 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 [2] (point-wise error control, average coverage control) and could be the object of ad-hoc optimization methods in future work.

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.

Acknowledgements

This work was funded by the French National Research Agency (ANR) through the grant CaMeLOt ANR-20-CHIA-0001-01 and the grant Pl@ntAgroEco 22-PEAE-0009. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

References

[1] Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. Smooth loss functions for deep top-k classification. In *ICLR*, 2018. 2, 7

[2] Evgenii Chzhen, Christophe Denis, Mohamed Hebiri, and Titouan Lorieul. Set-valued classification – overview via a unified framework. *arXiv preprint arXiv:2102.12318*, 2021. 2, 8

[3] 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. 3, 4, 5, 6

[4] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88:5–45, 2012. 3

[5] Christophe Denis and Mohamed Hebiri. Confidence sets with expected sizes for multiclass classification. *J. Mach. Learn. Res.*, 18:102:1–102:28, 2017. 2

[6] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009. 1

[7] 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. *Orphanet Journal of Rare Diseases*, 15(1):1–16, 2020. 7

[8] Matteo Fontana, Gianluca Zeni, and Simone Vantini. Conformal prediction: a unified review of theory and new challenges. *Bernoulli*, 29(1):1–23, 2023. 2, 3

[9] C. Garcin, A. Joly, P. Bonnet, A. Affouard, JC. Lombardo, M. Chouet, M. Servajean, T. Lorieul, and J. Salmon. Pl@ntNet-300K: a plant image dataset with high label ambiguity and a long-tailed distribution. In *NeurIPS Datasets and Benchmarks 2021*, 2021. 1, 3, 4, 7

[10] 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*, volume 162 of *Proceedings of Machine Learning Research*, pages 7208–7222. PMLR, 17–23 Jul 2022. 2, 7

[11] 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. *People and Nature*, 2023. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 7

[13] 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*, pages 8769–8778, 2018. 1

[14] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017. 7

[15] Faiza Khan Khattak. *Toward a Robust and Universal Crowd Labeling Framework*. PhD thesis, Columbia University, 2017. 3

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009. 3, 4, 7

[17] Maksim Lapin, Matthias Hein, and Bernt Schiele. Top-k multiclass SVM. In *NeurIPS*, pages 325–333, 2015. 1, 2

[18] 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. *Applications in plant sciences*, 8(6):e11365, 2020. 4

[19] Weiwei Liu, Haobo Wang, Xiaobo Shen, and Ivor W Tsang. The emerging trends of multi-label learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7955–7974, 2021. 3

[20] Titouan Lorieul. *Uncertainty in predictions of deep learning models for fine-grained classification*. PhD thesis, Université de Montpellier, Dec. 2020. 2, 3, 5, 8

[21] Titouan Lorieul, Elijah Cole, Benjamin Deneu, Maximilien Servajean, Pierre Bonnet, and Alexis Joly. Overview of geolifeclef 2022: Predicting species presence from multi-modal remote sensing, bioclimatic and pedologic data. In *CLEF 2022-Conference and Labs of the Evaluation Forum*, volume 3180, pages 1940–1956, 2022. 7

[22] Andre Machado, Rodrigo Veras, Kelson Aires, and Laurindo de Sousa Britto Neto. A systematic review on product recognition for aiding visually impaired people. *IEEE Latin America Transactions*, 19(4):592–603, 2021. 1

[23] 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. *Biodiversity Information Science and Standards*, 2019. 1

[24] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 3

[25] 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. *Int. J. Comput. Vision*, 115(3):211–252, 2015. 2, 3, 4, 7

[26] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008. 3

[27] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *J. Mach. Learn. Res.*, 14(Feb):567–599, 2013. 2

[28] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005. 3

[29] Forest Yang and Sanmi Koyejo. On the consistency of top-k surrogate losses. In *ICML*, volume 119, pages 10727–10735, 2020. 2

[30] Jiangbo Yuan, An-Ti Chiang, Wen Tang, and Antonio Haro. eproduct: A million-scale visual search benchmark to address product recognition challenges. *arXiv preprint arXiv:2107.05856*, 2021. 1

A. Hyperparameter sensitivity

We study the impact of the hyperparameters α and $|B|$ on average-K accuracy by conducting further experiments on CIFAR-100 (Section 6.2).

Figure 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 6 shows how average-K accuracy varies with the batch size for a model trained with ℓ_{CE} or $\ell_{\mathrm{AVG\text{-}K}}$. 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 $\ell_{\mathrm{AVG\text{-}K}}$ 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.

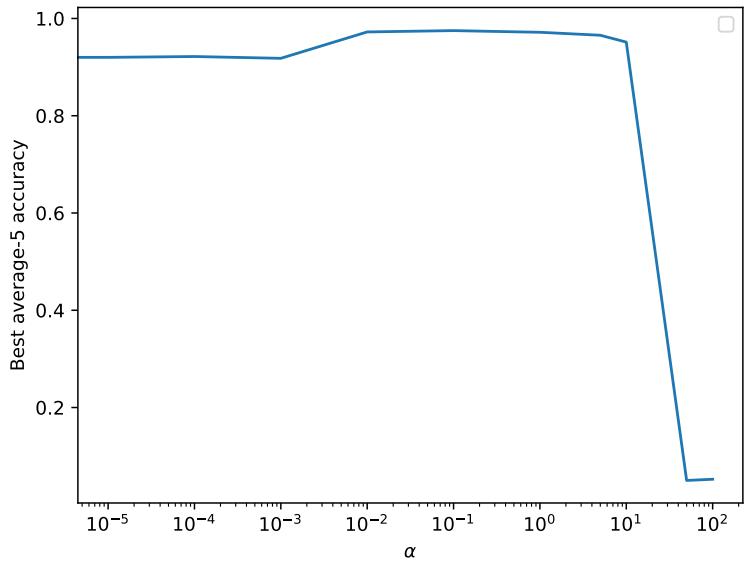


Figure 5: CIFAR-100 best validation average-5 accuracy for a DenseNet 40-40 trained with $\ell_{\mathrm{AVG\text{-}5}}$ for different values of α.

B. Experiments details

We report in Tables 3 and 4 the hyperparameters selected after grid search for all losses, for CIFAR-100 and Pl@ntNet-300K datasets respectively.

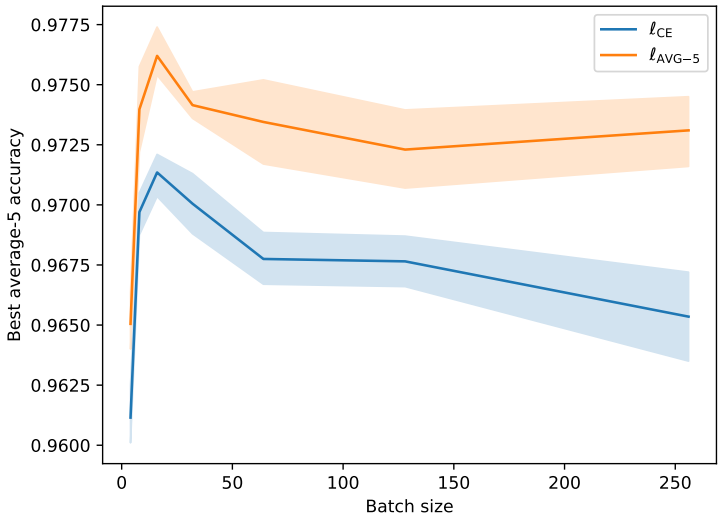


Figure 6: CIFAR-100 best validation average-5 accuracy as a function of batch size for a DenseNet 40-40 trained with $\ell_{\text{AVG-5}}$ or ℓ_{CE}. For a fair comparison we maintain the ratio of learning rate to batch size constant. The 95% confidence interval is represented.

loss	hyperparameters
ℓ_{CE}	-
$\ell_{\text{AVG-K}}$	$\alpha = 0.3$
ℓ_{AN}	-
ℓ_{EPR}	$\beta = 0.01$
$\ell_{\text{TOP-K}}$	$\epsilon = 0.2$
ℓ_{ROLE}	$\lambda = 0.0$, learning rate Θ: $\times 1.0$

Table 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_{EPR}	$\beta = 0.001$
$\ell_{\text{TOP-K}}$	$\epsilon = 1.0$

Table 4: Hyperparameters selected after grid search for the Pl@ntNet-300K experiments.