Predicting the Accuracy of a Few-Shot Classifier

Myriam Bontonou, Student Member, IEEE, Louis Béthune, and Vincent Gripon, Senior Member, IEEE

Abstract—In the context of few-shot learning, one cannot measure the generalization ability of a trained classifier using validation sets, due to the small number of labeled samples. In this paper, we are interested in finding alternatives to answer the question: is my classifier generalizing well to previously unseen data? We first analyze the reasons for the variability of generalization performances. We then investigate the case of using transfer-based solutions, and consider three settings: i) supervised where we only have access to a few labeled samples, ii) semi-supervised where we have access to both a few labeled samples and a set of unlabeled samples and iii) unsupervised where we only have access to unlabeled samples. For each setting, we propose reasonable measures that we empirically demonstrate to be correlated with the generalization ability of considered classifiers. We also show that these simple measures can be used to predict generalization up to a certain confidence. We conduct our experiments on standard few-shot vision datasets.

Index Terms—Few-Shot Learning, Generalization, Supervised Learning, Semi-Supervised Learning, Transfer Learning, Unsupervised Learning

I. INTRODUCTION

In recent years, Artificial Intelligence algorithms, especially Deep Neural Networks (DNNs), have achieved outstanding performance in various domains such as vision [15], audio [2], games [28] or natural language processing [3]. They are now applied in a wide range of fields including help in screening and diagnosis in medicine [6], object detection [39], user behavior study [18] or even art restoration [10] . . .

Designing a DNN consists in finding an adequate network architecture and training it to perform a given task using available labeled data gathered in a training set. In practice, there is a risk of overfitting, that is to say the model focuses on details of the training samples and is not able to generalize well to new ones. This is why the generalization abilities of trained models are often evaluated using another set of labeled samples called the validation set.

Problematically, stressing the generalization of a model using a validation set requires having access to a large quantity of labeled data. Yet annotating data typically costs money or even more inconvenient, in some cases, the acquisition of data is itself costly. An extreme case is when one has only access to a few labeled samples, referred to as few-shot in the literature. In such a case, trained models are likely to cause overfitting due to the small diversity of training data. The non-accessibility to a validation set is thus even more critical.

The problem of few-shot learning has known many contributions in the past few years. The core idea of most methods consists in exploiting data or knowledge gathered for other tasks. For example, methods that rely on transferring knowledge usually train a DNN on a first task using a huge amount of data. Then, a latent representation obtained through this DNN is used as a feature extractor for the few-shot task to be solved. The problem is that features have been trained for another task. Consequently, novel classes can divide a base class, or to the contrary, be a union of base classes. Even worse, the overlap may be ill-defined. Consequently, according to data and tasks, performances can range from very bad to very good. In practice, a question of paramount importance is then to be able to guess in which case we are.

In most few-shot learning works, the ability of a network to generalize to previously unseen inputs is not assessed because of the lack of a validation set. Often, hyperparameters are tuned using the test set, which is not a good practice since the test set should only be used after the solution has been designed to compare to other works.

In this work, we are interested in tackling the following problem: how can we estimate the generalization ability of a classifier in the context of transfer-based few-shot learning? We study three different settings related to few-shot: i) supervised, where we only have access to a few labeled samples, ii) semi-supervised, where we have access to both a few labeled samples and a set of unlabeled samples and iii) unsupervised, where we only have access to unlabeled samples. In all cases, we propose reasonable measures that we empirically demonstrate to be correlated with the generalization ability of trained classifiers. We also show that these simple metrics can be used to predict generalization up to a certain confidence.

The paper is organized as follows. In Section II we introduce the formalism and methodologies of few-shot learning, and the tasks we aim to solve. Section III is dedicated to a review of the related work. In Section IV we highlight the main issues arising in transfer-based few-shot learning, while in Section V we propose measures to predict the accuracy of a classifier trained with few labeled samples. Their efficiency is assessed in the experiments of Section VI. A summary of our results is presented in Section VII.

II. BACKGROUND

This section is intended as a reference for the rest of the article. It provides the formalism behind few-shot classification and ways of addressing it. The three settings we consider - supervised, semi-supervised and unsupervised, are detailed. The networks, classifiers and datasets used throughout the article, are also introduced.

Few-shot classification aims at learning to distinguish classes using only a few labeled samples. As such, there are

1Code at https://github.com/mbonto/fewshot_generalization
A. Few-shot classification: a transfer-based approach

The first step in transfer-based approaches consists in learning to extract feature representations from the abundant labeled data of the base classes. Then, a classifier is trained to associate the extracted representations of the few available labeled samples of the considered task with the corresponding labels. In the next paragraphs, we detail these steps.

1) Backbone training on base classes: Recall that we want to train a neural network using a large available dataset. This network is usually called backbone. At the end of the training, we obtain a network function \(g\), which can be written as \(g = c_{W_b} \circ f_\theta\), where \(c_{W_b}\) is a \(C_b\)-way classifier whose parameters are \(W_b\) and \(f_\theta\) is a convolutional Neural Network. Note that typically, \(c_{W_b}\) is the last layer of the neural network, so that \(f_\theta\) outputs the penultimate representation within the trained neural network when processing an input element.

The features in base classes are a prior on the features expected to be found in novel classes. Hence, if the features are relevant for classification tasks over base classes, we expect them to also be relevant for the classification of novel classes.

2) Feature Space: Once the backbone has been trained, we can generate the feature representations of the data used in the few-shot task. Denote by \(x\) a sample we are given. We obtain \(f = f_\theta(x)\) its corresponding feature extraction. We call data space the space where \(x\) has been drawn. Its extraction \(f_\theta(x)\) is part of the feature space.

More formally, let \(X\) be the data space in which lay the images. The training set induces a distribution \(P_X\) on this space. Let \(F\) denote the feature space (i.e the range of \(f_\theta\)). The feature extractor \(f_\theta: X \rightarrow F\) induces a distribution \(P_{F|X}\).

To solve the given task, the principle is to train a classifier to associate the extracted features with the corresponding labels. Denote by \(c_{W_n}\) this classifier, then samples are classified using the composition: \(c_{W_n} \circ f_\theta\), where \(c_{W_n}\) is a \(C_n\)-way classifier.

Let \(Y\) denote the space of class labels, a finite set containing all possible labels for the task. The classifier on top of the backbone induces a distribution \(P_{Y|F}\). The predicted label associated with the features of a data sample \(x \in F\) is denoted \(\hat{y}_x\) while the true label is \(y_x\).

B. Studied problems

We consider three different settings: a supervised setting, an unsupervised setting and a semi-supervised setting. The data space \(X\) carries very little usable information, so from now on, all the examples are assumed to come from \(F\) with distribution \(P_{F|X}\), through the use of the above-mentioned backbone. We now introduce the three problems in details.

1) Supervised setting: We consider \(N\)-way \(K\)-shot tasks. The number \(N\) refers to the number of classes to discriminate from, and \(K\) to the number of labeled samples we are given for each class. Denote by \(x_{ij}\) the \(i\)-th labeled element of class \(j\), then the training set can be written as:

\[
S^{\text{supervised}} = \{(x_{ij}, y_j) | 1 \leq i \leq K, 1 \leq j \leq N\}.
\] (1)

2) Unsupervised setting: We consider \(N\)-way \(Q\)-query tasks. We are given \(Q\) unlabeled samples for each of the \(N\) considered classes. The training set can be written as:

\[
S^{\text{unsupervised}} = \{(x_{ij}, \bot) | 1 \leq i \leq Q, 1 \leq j \leq N\}.
\] (2)

Here, \(\bot\) refers to the fact we do not know what the labels are.

3) Semi-supervised setting: This setting is the combination of the previous two. We consider \(N\)-way \(K\)-shot \(Q\)-query tasks. Some training samples are labeled, others are not. The training set is:

\[
S^{\text{semi}} = S^{\text{supervised}} \cup S^{\text{unsupervised}}.
\] (3)

This lastly considered case is often the most desirable, since we can benefit from both the unlabeled and labeled samples when training the classifier.
Remark: In both unsupervised and semi-supervised settings, although we do not have access to the labels during training, we know that each class is equally present. This is the methodology considered in several research papers [12], [16] for evaluation purposes. We also include, in Section V, an additional experiment where the distribution of classes is unbalanced.

C. Experimental setting

In this section, we introduce the considered datasets, backbones and classifiers.

1) Datasets: We consider two datasets. The first one is mini-ImageNet [33]. It has been generated from the bigger ImageNet database [25]. It is split into 64, 16 and 20 classes, in which 600 images are available. The first split is used to train the backbone, the second to validate its generalization ability and the third one to generate the few-shot tasks. The second dataset is tiered-ImageNet [24]. The splits contain 351, 97 and 160 classes, with roughly about 1000 samples each. It is also extracted from ImageNet. The interest of tiered-ImageNet is that the semantic of classes has been studied with WordNet [21] to ensure that the considered splits contain semantically different classes. In both datasets, as in numerous studies [34], the images are resized to $84 \times 84$ pixels.

When generating a few-shot task, $N$ classes are uniformly drawn at random in the last introduced split. The $K$ and $Q$ samples to generate from each class are uniformly drawn without replacement. To assess the generalization performance in the supervised setting, we usually measure the classification accuracy that is achieved on 50 samples uniformly drawn from the remaining items for each considered class (that is to say items that were not drawn to be part of the $K$-shot). In the unsupervised and semi-supervised settings, the performances are measured on the $Q$ samples per class.

2) Backbones: We consider three feature extractors. The first one is a Wide Residual Network [38] of 28 layers and width factor 10 described in [19]. It has been trained on mini-ImageNet with a classification loss (classification error), an auxiliary loss (self-supervised loss) and fine-tuned using manifold mix-up [32]. Its results are among the best reported in the literature. In the following, we denote it by widenet. For more extensive benchmarks, we also use a DenseNet [13] trained on mini-ImageNet, and the same DenseNet trained on tiered-ImageNet. Both are described in [34]. They are respectively referred to as densenet-m and densenet-t in the following. As advised in the original papers, we divide all feature vectors by their $L_2$-norm. Given $f \in \mathcal{F}$, $f \leftarrow \frac{f}{\|f\|_2}$.

3) Classifiers:

a) Supervised setting: A classifier allowing nearly always to get the best performances in the supervised setting is the Logistic Regression (LR). Given $d$ the number of dimensions of the feature vectors, a weight matrix $W \in \mathbb{R}^{d\times N}$, the matrix containing the features of all data samples $F \in \mathbb{R}^{(NK)\times d}$, the output of the LR $P \in \mathbb{R}^{(NK)\times N}$ is:

$$P = \text{softmax}(FW),$$

where $p[i,c]$ is the probability that the sample $i$ belongs to class $c$. Weights are learned using backpropagation to minimize a cross-entropy loss. It constrains the novel classes to be separated by hyperplanes in the feature space $\mathcal{F}$. A stronger classifier would allow the examples to lay on a manifold with more complex shapes, making clustering more challenging. If not specified otherwise, LR is trained on 50 epochs with Adam optimizer [14], a learning rate of 0.01 and a weight decay of $5e^{-6}$.

b) Semi-supervised setting: One of the state-of-the-art classifiers is an adapted Logistic Regression [12]. The features extracted from a backbone are diffused through a cosine similarity graph $\mathcal{G}$ before being processed by a usual LR.

Definition 1 (Cosine similarity): Given $f_i, f_j \in \mathcal{F}$, the cosine similarity function is defined as:

$$\cos(f_i, f_j) = \frac{f_i^\top f_j}{\|f_i\|_2 \|f_j\|_2}. \quad (5)$$

As in all backbones we use, the features are extracted after a ReLU function, all vectors in $\mathcal{F}$ contain non-negative values. Consequently, the output of the cosine similarity function ranges from 0 to 1.

Definition 2 (Cosine similarity graph): A cosine similarity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ consists in a set of vertices $\mathcal{V}$ connected by a set of edges $\mathcal{E}$. The weights of the edges are stored in the adjacency matrix $W$. Given two vertices $i, j$ and their feature vectors $f_i, f_j$,

$$W[i,j] = \begin{cases} \cos(f_i, f_j) & \text{if } \{i, j\} \notin \mathcal{E} \\ 0 & \text{otherwise} \end{cases}. \quad (6)$$

We consider a cosine similarity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ (see Definition 1 and 2). After removing self-loops, we only keep the $k$-th largest values on each row and we normalize the resulting matrix as follows:

$$E = D^{-\frac{1}{2}} WD^{-\frac{1}{2}}, \quad (7)$$

where $D$ is the diagonal degree matrix defined as:

$$D[i,i] = \sum_j W[i,j]. \quad (8)$$

Given $F \in \mathbb{R}^{(NK+NQ)\times d}$ the matrix containing the features of all data samples and $I$ the identity matrix, the new features are obtained by propagating the extracted features as follows:

$$F_{\text{diffused}} = (\alpha I + E)^k F. \quad (9)$$

In the reference paper [12], $\alpha, \kappa$ and $k$ are hyperparameters. The best ones found on 5-way 5-shot 15-query tasks on mini-ImageNet are $\alpha = 0.75$, $k = 15$ and $\kappa = 1$. We use these hyperparameters in all our experiments.

Finally, a LR is trained on the diffused features $F_{\text{diffused}}$ with the same parameters as in the supervised setting.
c) Unsupervised setting: The unsupervised setting is less studied in the few-shot literature. We hypothesize that, when features are well adapted to a $N$-way task, each class is associated with a cluster in the feature space. In that case, a standard clustering method consists in using a $N$-means algorithm. We use the implementation of the algorithm with default values as implemented in scikit-learn [23]. In order to compare results with the semi-supervised setting in a fair manner, we also propagate the features extracted from backbones through a cosine similarity graph as we explained it in the semi-supervised setting. To evaluate the quality of the clustering, we compute its Adjusted Rand Index (ARI). This index ranges from 0 to 1, 1 meaning that the data samples are exactly clustered according to their labels, and 0 that the clustering is at chance level.

In the next section, we discuss related work and introduce ours in a broader context.

III. RELATED WORK

In machine learning, there is an increasing interest for learning a task with few samples. The literature on this subject is referred to as few-shot learning, where the authors can mean either the supervised setting or the semi-supervised setting is considered. In this section, we detail some of the most important articles of this field.

A. Few-shot learning

As training a Deep Neural Network (DNN) on few data samples from scratch typically leads to overfitting, several learning strategies have been developed over the past years. All these strategies share the idea of building a general-purpose representation (using a backbone network) of data samples. For instance, in image classification, some knowledge is retrieved from a rich database containing training classes, called base classes. Then, this knowledge is transferred to perform a new task on novel classes.

a) With meta-learning: A first group of strategies uses meta-learning. It consists in using entire tasks as training examples. Some optimization-based works propose to learn a good initialization of the weights of the DNN over several training tasks, so that a new task can be learned with only a few gradient steps [9], [26]. In metric-based works [22], [29], [30], [33], [37], the idea is to learn to embed the data samples in a space where classes are easily separable. Thus, once a new task occurs, the features of the novel samples are projected into this embedding (without any learning) and a simple classifier is trained to recognize the novel classes from these features. As the number of parameters to learn is reduced, the risk of overfitting is lower. There are many variants in the literature. For instance, in [29], they assume that it exists an embedding space where each class is represented by one point. Thus, a DNN is trained over several training tasks to work with a distance-based classifier, in which each class is represented by the average of its projected data samples. When a new task comes, the representations of the samples are extracted from the DNN, and the labels of the query samples are attributed according to the class of the closest representative.

b) Without meta-learning: In a recent line of work, some methods do not focus on learning a relevant embedding from training tasks but on learning a relevant embedding from a single classification task involving all training classes at once [7], [19], [31], [34]. First, a DNN is trained to minimize a classification loss on base classes. A regularization term such as self-supervision [19], [31] or Manifold Mixup [19] is sometimes added to the loss to learn more robust features. Then, the features of the samples of the few-shot task are extracted from the DNN (often using the features of its penultimate layer, just before the classifier). Finally, a simple classifier, such as a Nearest Class Mean [34] or a Cosine Classifier [19], is trained on the extracted features to distinguish between classes. In [34], the authors show that simple transformations, such as normalizing extracted features with their $L_2$-norm, help the classifier generalizing better. Using self-supervision and Manifold Mixup, the article [19] achieves state-of-the-art performances on benchmark datasets. That is why, in this article, we allow to restrict our study to few-shot learning solutions based on pretrained feature extractors.

B. Generalization to new classes

In transfer-based few-shot learning, the challenge is to learn representations on training classes, which are suitable for novel classes. Indeed, the generalization ability of a classifier is linked to the distribution of the representatives of the data samples in the feature space. However, it is not easy to estimate whether the learned embedding space suits novel classes well.

a) Learning diverse visual features: The generalization ability of the classifiers depends on the relevance of the extracted features for a new task. Inspired by works in deep metric learning, the authors of [20] propose to learn representations capturing general aspects of data. They optimize a DNN to perform a range of tasks enhancing class-discriminative, class-shared, intra-class and sample-specific features. Although they do not apply their method to few-shot tasks, it could help improving the generalization. Similarly, self-supervised learning and Manifold Mixup used in [19] improve the accuracy on few-shot tasks. Another way to learn richer representations is to use additional unlabeled samples.

b) Using additional unlabeled data samples: When unlabeled samples are available, they can be used to infer more adapted representations of data samples to distinguish between novel classes. In the literature, two settings are studied. In one of them, the unlabeled samples are the query samples on which the accuracy of the classifier is evaluated. This is the setting we consider in this article. In the other setting, the unlabeled samples are just additional samples and they are not used to test the classifier. The authors of [16] consider both settings. They look for a linear projection which maximizes the probability of being in the correct class. More precisely, an unsupervised low-dimensional projection (PCA or ICA) is first applied on the features to reduce their noise. Then, data samples are clustered using two possible methods: a Bayesian $K$-Means or a Mean-Shift approach followed by a NCM classifier. In [12], the features of the data samples are diffused though a similarity graph computed from the few-shot
samples and from the unlabeled samples before being used in a classifier. As these works use additional information, the generalization performances are increased.

c) Learning good representations: Learning efficient representations has always been a concern for deep learning [4]. Invariant Risk Minimization [1] and $\nu$-Information [36] have been proposed as theoretical frameworks to detail the properties a good representation should exhibit when connected to a (mostly linear) classifier. Other works focus on maximization of Mutual Information (following InfoMax principle) such as Deep Infomax [11]. Losses (like in [20] or in [35]) are designed to enforce some geometry in latent space based on similarity measures (such as sharing the same label). Robust few-shot learning for user-provided data [17] is proposed to handle outliers within training samples.

d) Evaluating the generalization ability: The generalization ability of a few-shot solution can be improved by designing more relevant representations of data. However, this ability is ill-evaluated. In standard deep learning, the generalization performance is computed on a validation set. Here, we do not have enough samples to afford such a procedure. Thus, the question of interest in this study, which has not been handled so far in the literature, is not how to improve the generalization performance but really how to evaluate it.

IV. CASE STUDY ON MINI-IMAGENET

The current case study section has been designed to outline how difficult it is to predict the generalization ability of a classifier on a few-shot task. We begin with a motivating observation. With mini-ImageNet, we can generate many 5-way 5-shot tasks. As mini-ImageNet contains hundreds of samples per class, we also generate test sets for all tasks. Solving them using a combination of wideresnet with LR, we observe that the performances on the test sets vary from 55.6% to 98% on a sample of 10,000 runs.

To better understand the reasons for these variations, we conduct an experiment, detailed in subsection IV-A. In brief, the accuracy on a task mainly depends on two variables, the distribution of classes with respect to each other and the random selection of labeled samples within classes. On mini-ImageNet, we show that the impact of the choice of shots is minor compared to the one of the classes. In subsection IV-B, we propose an interpretation of the difficulty induced by the choice of classes.

A. Identifying the source of overfitting

In order to look for the origin of the difficulty of a few-shot task, we propose the following experiment on the mini-ImageNet dataset. We consider 5-way 5-shot tasks and we extract the features of the samples with wideresnet. When we generate a task, we fix either the choice of classes or the choice of shots within classes. While varying the other variable, we look at the standard deviation of the accuracy of the task. If the standard deviation is low, we consider the variable has little impact.

First, we compute the standard deviation of the accuracy over random tasks (see Algorithm 1). Second, we explore the impact of the choice of shots within classes on the accuracy of a task. We compute the standard deviations of the accuracy while fixing the classes (see Algorithm 2). Third, we explore the impact of the choice of classes by computing the standard deviations of the accuracy while fixing the shots within each class (see Algorithm 3).

The standard deviation of the accuracy over random 5-way 5-shot tasks is 5.85%. The average of the standard deviations over tasks with fixed classes and random shots is 2.02%. Its standard deviation is 0.35%. The average of the standard deviations over tasks with random classes and fixed shots is 4.37%. Its standard deviation is 0.61%. We observe a higher variability within the choice of classes than within the selection of labeled samples within classes.

In this subsection, we considered both variables independent. However, in reality, they are not. If we consider the

---

**Algorithm 1:** Standard deviation of the accuracy

```python
perf ← empty list
for run = 1 → 500 do
    Select randomly 5 classes and 5 shots within classes.
    Generate a test set with 50 other samples per class.
    Train a LR to classify the 5 classes from the features.
    Append the LR accuracy on the test set to perf.
end for
return standard_deviation(perf)
```

**Algorithm 2:** Standard deviation while fixing classes

```python
stds ← empty list
for run = 1 → 500 do
    Select randomly 5 classes.
    perf ← empty list
    for run = 1 → 500 do
        Select randomly 5 shots within each class.
        Generate a test set with 50 other samples per class.
        Train a LR to classify the 5 classes from the features.
        Append the LR accuracy on the test set to perf.
    end for
    stds.append(standard_deviation(perf))
end for
return average(stds)
```

**Algorithm 3:** Standard deviation while fixing shots

```python
stds ← empty list
for run = 1 → 500 do
    Select randomly 5 shots within each class.
    perf ← empty list
    for run = 1 → 500 do
        Select randomly 5 classes and retrieve their 5 shots.
        Generate a test set with 50 other samples per class.
        Train a LR to classify the 5 classes from the features.
        Append the LR accuracy on the test set to perf.
    end for
    stds.append(standard_deviation(perf))
end for
return average(stds)
```
closest classes, the choice of shots is much more important. The average of the standard deviations over tasks with fixed classes and random shots is 2.26%. Its standard deviation is 0.38%. The average of the standard deviations over tasks with random classes and fixed shots is 3.97%. Its standard deviation is 0.31%. Similarly, if classes are far away, the choice of shots does not have an influence anymore.

In the next section, we propose to unveil the inherent difficulty of some tasks. To understand the impact of the choice of classes on the generalization, we propose to study the distribution of classes in the input space of the classifier (which we called the feature space). Thus, we look at the topological space of data samples after they went through the feature extractor.

**B. Measuring class confusion**

We show in the previous subsection that, for a fixed backbone, the variability in accuracy seems to be more dependent on the choice of classes than on the choice of examples inside those classes. Then arises the question: **what are the pairs of classes \( \{A, B\} \) hard to distinguish for a given backbone?**

We refer to those pairs of classes as **hard tasks**. The natural follow up being: **are the hard tasks the same for different backbones?**

1) **Graph based measure of domain overlapping**

To answer these questions, we need to measure how much the domain of different classes overlap. In this subsection, we introduce such measure, based on graph similarity. Given two classes \( A \) and \( B \), we build a cosine similarity graph \( G = (\mathcal{V}, \mathcal{E}, \mathbf{W}) \) of the examples in the latent space. (see Definition 2). The set of vertices gather all samples from both classes: \( \mathcal{V} = \mathcal{X}_A \cup \mathcal{X}_B \). After removing self-loops, we only keep the \( |\mathcal{V}| \) heavier edges among the possible ones, i.e. we have \( |\mathcal{E}| = k|\mathcal{V}| \).

The densely connected graph corresponds to \( k = |\mathcal{V}| \) whereas \( k = 0 \) leads to isolated vertices. For an appropriate choice of \( k \) somewhere between those two extremes, the resulting graph is sparse and naturally organized into communities with high intra-connectivity. Here, we chose \( k = 20 \).

By applying **Louvain’s communities detection algorithm** [5] we detect the cluster \( C_i \in \mathcal{C} \) associated with the example \( i \in \mathcal{V} \). If the classes \( A \) and \( B \) are well separated, we expect those communities to contain elements of a single class. Conversely, if the domains of classes overlap, those communities will contain different labels. In few-shot setting, due to the lack of examples and the high dimensional nature of data, most algorithms (supervised or unsupervised) rely on the hypothesis that examples in the same neighborhood must share the same label. Hence, we make the hypothesis that a classifier can predict the true label \( y \in \{A, B\} \) of the example \( i \) knowing its community \( C_i \). In this case, the **cross entropy** \( CE \) of any classifier \( c_{W_n} \) against the true distribution \( P_{Y|C} \) is lower bounded by the conditional entropy:

\[
CE(P_{Y|C}, c_{W_n}) \geq H(Y|C) = E_{C\sim C} (H(p_A(C))) = E_{Y\sim Y} (H(p_A(C_i))),
\]

where \( Y \) (resp. \( C \)) is the random variable of labels (resp. communities) under uniform sampling of vertices in \( i \in \mathcal{V} \). This lower bound can be computed empirically from the data using the true labels. The frequency \( p_A(C_i) \) of labels \( A \) in communities \( C_i \) is used to compute the binary entropy:

\[
H(p_A(C_i)) = - p_A(C_i) \log_2(p_A(C_i)) - (1 - p_A(C_i)) \log_2(1 - p_A(C_i)),
\]

which reaches its maximum when both classes are equally present in the community, and a minimum when the community only contains one class. Its weighted average over communities is the lower bound. The stochasticity of Louvain’s algorithm is counter-balanced by an average over 5 runs:

\[
S(X_A,X_B) = \frac{1}{5} \sum_{r=1}^{5} H_r(Y|C).
\]

2) Results: The scores \( S(X_A,X_B) \) are computed for each pair of classes \( \{A, B\} \) among the 20 classes of mini-ImageNet. The features of the images are extracted with **densenet-m**. The higher the score, the greater the overlap between classes. By keeping the most significant scores, we build the graph of Fig. 1(a).

We notice that the mammals are clustered, reflecting an important overlapping between the corresponding classes. Without prior knowledge on the concept of mammal, the backbone suffers from this confusion. They are all connected to each other, meaning they are all sharing some features: it is an example of class confusion. The confusion between **Lion** and **African hunting dog** may involve the desert background.
The Vase class is an example of multiple class representatives: the domain of this class overlaps some other such as Mixing bowl, Cuirass or Hourglass. But there is no transitivity: Mixing bowl and Cuirass are not very similar to each other. The features of Vase are split among different other novel class. Such phenomenon can also be observed with Nematode, Ant and King crab.

Moreover, the score $S(x_A, x_B)$ is positively correlated with the average error of a logistic regression. To show it, we uniformly sample 5 classes and compare the sum of $S(x_A, x_B)$ over those five classes with the generalization error in a supervised setting over 5-way 5-shot tasks. The obtained points are depicted in Fig. 3. The Pearson correlation coefficient is 0.84, estimated with 10,000 runs. We choose the 595 remaining examples for test set, since it gives a more reliable estimate of the performance of the classifier.

3) Influence of the backbone: A reasonable question is whether the obtained results are specific to the considered backbone. This is why we proceed to the same evaluation with wideresnet and we obtain the graph in Fig. 1(b).

The two graphs are very similar. On the complete graph (without cutting lighter edges), the correlation edgewise is 0.93. This high correlation encourages us to think that hard tasks are mostly independent of the backbone and are strongly influenced by the base classes used for backbone training. We wonder if class confusion in novel classes is induced by class confusion with base classes. Said otherwise: are novel classes confused with base classes sharing similar features?

4) Novel classes are confused with base classes: To answer this question, we build similarity graphs like the ones of the previous section, involving both base classes and novel classes for densenet-m. We enforce the graph to be bipartite, and we only keep the heavier edges to increase readability (see Fig. 2). We observe mainly three phenomena.

a) Perfect matching between novel and base class: It is the case for the pairs bookshop and tobacco shop, and for ant and harvestman. In this situation, the features of the novel class are entirely determined by the features of one base class, which are not split among other novel classes. The two novel classes bookshop and ant are not part of a cluster nor a hub in Fig. 1(a). Hence, it seems to be the most desirable case for few-shot learning.

b) Split of base classes: In this situation, a base class provides features that are shared by many novel classes. It is the case for saluki (a specie of dog) which, according to intuition, shares features with two other species of dog (golden retriever, malamute). Interestingly, the dogs are organized into a hub in Fig. 1(a). Vase, hourglass and cuirass are also organized into a hub, and they appear to share features with cocktail shakers. Hence, the split of base classes seems to be an important cause of the apparition of hubs in Fig. 1(a).

c) Split of novel classes: In this situation, the domain of a novel class overlaps the domain of multiple base classes. The most notorious case is the electric guitar, which is split among five base classes, including stage (which may contain image of guitar), oboe (another music instrument), reel (many of which are from fishing rods), holsters or clogs. It is also the case of mixing bowls, confused with frying pans and woks, or hourglass with cocktail shakers and chimes. It seems that the novel classes that are split are involved in more conflicts in Fig. 1(a).

C. Conclusion

We pointed out the problem of the confusion of classes in the feature space when using a transfer-based few-shot solution. The difficulty arises from the overlapping of domains of novel classes that make few-shot approaches inefficient. We developed a measure of domain overlapping (Fig. 1), and we established its correlation with the accuracy of naive LR (Fig. 3). The domain overlapping in latent space depends more on the base classes than on the backbone and its training procedure. It is explained by the fact that novel classes share features with base classes (Fig. 2).

This confusion leads the classifier to overfit on the few available labeled samples. As all labeled samples are used during the training process, the generalization performance of the classifier cannot be checked on another set of samples. The challenge of the following sections consists in predicting the generalization ability of a classifier without having access to more labeled data samples.

V. Methods

Let us remind the setting of the study. Having access to a feature extractor trained on a rich database, a classifier learns to distinguish between new classes using few labeled samples. We aim at quantifying how well the trained classifier generalizes to unseen data.

When performing benchmarks, authors usually have access to more labeled data through a test set. However, in practical applications, when learning from few labeled samples, it is not simple to generate such a set. In this section, we propose several metrics that do not require a test or a validation set, and aiming at quantifying how well a model generalizes to unseen data. A recap of studied metrics is given in Table 1.
TABLE I: Table presenting several settings encountered while learning with few labeled samples, and summarizing considered solutions. Solutions are metrics designed to quantify how well a trained model generalizes to unseen data.

| PROBLEMS                 | Supervised N-way K-shot | Semi-supervised N-way K-shot Q-query* | Unsupervised N-way Q-query* |
|--------------------------|-------------------------|---------------------------------------|-----------------------------|
| Using available labels and features of data samples | ✓ | ✓ | × |
| Training loss of a Logistic Regression | ✓ | ✓ | ✓ |
| Relative distances between labeled samples | ✓ | ✓ | × |
| Confidence in the output of the classifier | ✓ | ✓ | × |
| Using only data relationships | ✓ | ✓ | ✓ |

Query samples are accessible during training without their labels.

In the following subsections, we introduce measures for each considered setting that we prove to be correlated with the generalization performance in the next section.

A. Metrics defined on supervised inputs

Recall that the classifier we use in the experiments is a Logistic Regression (LR). We propose two metrics to estimate how well the trained LR generalizes to unseen data. The first one consists in using the obtained LR training loss at the end of the training process. The second one is based on the relative distances between labeled data samples, and is therefore agnostic of the choice of the LR as classifier.

**LR training loss:** The LR is trained to minimize the cross-entropy loss, see Definition [3]. During training, this loss is supposed to converge to zero. Assuming that harder the task is, slower the convergence is, the value of the loss at the end of the training should give some insights about the difficulty of the task. Thus, we decide to explore its ability to estimate how easy the generalization to new data samples will be.

**Definition 3 (LR training loss):** Given \( y_{ic} \), the number (0 or 1) indicating if the label of the data sample \( i \) is \( c \) and \( p_{ic} \), the output of the LR indicating the probability of \( i \) being labeled \( c \), the loss is defined as:

\[
\text{LR training loss} = -\frac{1}{NK} \sum_{i=1}^{N} \sum_{c=1}^{K} y_{ic} \log p_{ic} .
\] (13)

**Similarity metric:** Recent works [12], [34] have shown that state-of-the-art performances can be achieved with a decision process that ultimately consists in comparing distances to a centroid defined for each class. Thus, a natural indicator of how easy such a clustering can be performed is to compare the intra-class similarity to the inter-class similarity. In our case, we choose the cosine similarity as the feature vectors are projected onto the unit sphere. We detail the metric we propose by first introducing the notions of intra-class and inter-classes similarities respectively in Definition [4] and [5]. Then the proposed metric is written in Definition [6].

**Definition 4 (Intra-class similarity):** The cosine similarity within a class \( c \) is:

\[
\text{intra}(c) = \frac{2}{K(K-1)} \sum_{i} \sum_{j \neq i, y_i = c, y_j = c} \cos(f_i, f_j) .
\] (14)

If a class \( c \) only contains one shot, we set \( \text{intra}(c) = 1 \).

**Definition 5 (Inter-classes similarity):** The cosine similarity through classes \( c \) and \( \hat{c} \) is:

\[
\text{inter}(c, \hat{c}) = \frac{1}{K^2} \sum_{y_i = c} \sum_{y_j = \hat{c}} \cos(f_i, f_j) .
\] (15)

**Definition 6 (Similarity metric):** The proposed similarity metric is:

\[
\text{similarity} = \frac{1}{N} \sum_{c=1}^{N} \left( \text{intra}(c) - \max_{c \neq \hat{c}} \text{inter}(c, \hat{c}) \right) .
\] (16)

B. Metrics defined on unsupervised inputs

In the unsupervised setting, we use the N-means algorithm. Again, the goal is to estimate the quality of the obtained clustering. To this end, we consider two metrics. The first one is based on a relative similarity measure between clusters. The other one is an indirect measure of component connectivity in a graph whose vertices are the considered samples and edges represent the similarity between those samples.

**Davies-Bouldin score after N-means:** Assuming the data samples within classes are similar enough, we expect each learned cluster to represent a class. A measure of relative similarity within clusters and between clusters, such as the classical Davies-Bouldin (DB) score [8], gives an insight about the difficulty of the clustering. Consequently, it may measure how easy it is to generalize to new samples. In Definition [7] we detail the Davies-Bouldin (DB) score. Lower is the score, better is the clustering. It varies between 0 and +∞.

**Definition 7 (Davies-Bouldin score):** Denote the centroid of a cluster \( C_c \), \( \mu_c \), such that \( \mu_c = \frac{1}{|C_c|} \sum_{i \in C_c} f_i \). The average distance between samples and the centroid of their cluster is:

\[
\delta_c = \frac{1}{|C_c|} \sum_{i \in C_c} \|f_i - \mu_c\|_2 .
\] (17)

Then, the DB score is given as:

\[
\text{DB score} = \frac{1}{N} \sum_{c=1}^{N} \max_{c \neq \hat{c}} \left( \frac{\delta_c + \delta_{\hat{c}}}{\|\mu_c - \mu_{\hat{c}}\|_2} \right) .
\] (18)
**Laplacian eigenvalues:** Consider a graph where each vertex represents a data sample and edges are weighted according to a similarity between these samples. In a perfect case where samples from distinct classes are very dissimilar, it is expected that this graph yields at least as many connected components as the number of classes in the considered problem. A measure of the fact a graph contains at least $N$ connected components is given by the amplitude of the $N$-th lower eigenvalue of its Laplacian \( \mathbf{L} \). See Definition 8.

Definition 8 (\(N\)-th eigenvalue): We consider the graph \( \mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W}) \) where \( \mathcal{V} \) is the set of data samples. The adjacency matrix \( \mathbf{W} \) is obtained by first considering the cosine similarity between these samples, removing self-loops, and keeping only the \( k \)-th largest values on each line/column. See Definition 2 for more details. The Laplacian of the graph is given by \( \mathbf{L} = \mathbf{D} - \mathbf{W} \), where \( \mathbf{D} \) is the degree matrix of the graph: \( \mathbf{D}_{ii} = \sum_{j=1}^{NQ} \mathbf{W}_{ij} \). The measure we consider is the amplitude of the \( N \)-th lower eigenvalue of \( \mathbf{L} \).

**C. Metric defined in the semi-supervised setting**

Consider we have access to a LR classifier and unlabeled inputs. We propose a metric based on the confidence of the LR decision on the unlabeled samples. The confidence can be obtained by looking at the distance between the provided output and a one-hot-bit encoded version of this output.

In more details, for each unlabeled sample, the classifier outputs the probability it belongs to a particular class. As we do not know the label of the sample, we cannot look at the probability the classifier gives to its real class. However, we propose to report the maximal probability the classifier attributes to the classes (see Definition 9). If the maximal probability is far from one, we can interpret it as the classifier is unsure of its output. So, lower the maximal probability is, harder the task could be for the considered sample.

Definition 9 (LR confidence): Let \( p_{ic} \) denote the probability that the data sample \( i \) is labeled \( c \).

\[
\text{LR confidence} = -\frac{1}{NQ} \sum_{i=1}^{NQ} \log \max_c(p_{ic}).
\] (19)

In the next section, we empirically measure how correlated are the proposed measures with generalization performance. We also propose to predict the latter from the former.

**VI. EXPERIMENTS**

In this section, we evaluate the interest of metrics proposed in Section IV. We perform experiments on data samples coming from mini-ImageNet with features extracted using wideresnet and on data samples from tiered-ImageNet with features extracted using densenet-t. In the unsupervised and semi-supervised settings, let us recall that the extracted features are diffused through a similarity graph before being used. More details on the datasets and the backbones are in Section III.

**A. Supervised setting**

We begin with the supervised setting. We propose to study the linear correlation between the metrics defined on supervised inputs computed on a \( N \)-way \( K \)-shot task and the accuracy of the LR on a test set. The test set gathers 50 unseen data samples per class. We also look at the correlations obtained with the metrics defined on unsupervised inputs, considering all training samples are unlabeled. In Fig. 4 we perform experiments on mini-ImageNet ((a) and (b)) and on tiered-ImageNet ((c) and (d)). In (a) and (c), we consider 5-way tasks and depict the evolution of the correlation as a function of the number of shots per class. In (b) and (d), we consider 5-shot tasks and make the number of classes vary. Note that in 1-shot tasks, as the DB score is always 0, we do not know the label of the sample, we cannot look at the distance between the provided output and a one-hot-bit encoded version of this output.
not report a correlation measure. In Appendix A, the average accuracies of the LR for each type of task are given.

**Observations:** In all experiments, we observe that the metrics adapted to the supervised setting (i.e. LR loss and similarity) perform better than the metrics designed for an unsupervised setting. The best correlation is always obtained with the LR loss. About the LR loss and the similarity, the more shots there are, the better the correlations are. The more ways there are, the worse the correlations are on mini-ImageNet. On tiered-ImageNet, they do not vary.

**Discussion:** The metrics designed for a supervised setting overcome the ones designed for an unsupervised setting. Indeed, the supervised metrics exploit an additional information given by the labels. Moreover, the unsupervised metrics are here used on very few data samples. In the next subsections, we show the results they can obtain using more unlabeled samples. The number of shots and ways influence the complexity of the tasks. Higher the number of ways is or lower the number of shots is, harder the task becomes. The linear correlation seems to be lower when the tasks are harder.

We investigate what happens in Fig. 5. We generate two plots using data samples from mini-ImageNet. Each point represents a task, with the LR loss on the X-axis and the accuracy on the Y-axis. In (a), 5-way 5-shot tasks are considered. In (b), 5-way 1-shot tasks. In 5-way 5-shot, the relation between both variables is rather linear. Without surprise, in 5-way 1-shot, the LR loss is less representative of the accuracy. With only one sample per class, it is very hard to detect the problematic tasks.

**B. Unsupervised setting**

In this setting, we consider N-way Q-query tasks. We study the linear correlations between the metrics and the adjusted rand index (ARI) of a N-means algorithm. The ARI is computed on the unlabeled samples available when training the N-means algorithm. In Fig. 6, we perform experiments on mini-ImageNet ((a) and (b)) and on tiered-ImageNet ((c) and (d)). In (a) and (c), we consider 5-way tasks. The number of queries varies. In (b) and (d), we consider 35-query tasks. The number of classes N varies. In Appendix A, the average ARI of the N-means for each type of task are given.

**Observations:** In all experiments, we observe that the DB-score is the best. The more queries there are, the better the correlation is, although there is a threshold after 20 queries per class. The correlation with the N-th eigenvalue slightly decrease but this is an artifact due to the construction of the graph. We propose, in Appendix B, an experiment showing the influence of the number of nearest neighbors kept in the graph used to compute the eigenvalue. We also observe that the more classes there are, the lower the correlation becomes.

**Discussion:** As the DB-score is more aligned to the N-means algorithm, its correlation with the ARI is higher. We notice that 20 samples per class are sufficient to increase the correlation up to 0.64 on mini-ImageNet and up to 0.72 on tiered-ImageNet. It seems that using more samples is useless. As the complexity of the tasks increases with the number of ways, our metrics are less representative of the problems.

In this section, we do not have access to the labels of the samples during the training but, we make sure that each class contains as many samples. In subsection VI-E, we explore the impact of an unbalanced distribution. We also propose an additional experiment to see if the N-th eigenvalue is really the one giving the best correlation among all other eigenvalues.

In the next subsection, we wonder to what extent the linear correlations can be increased in a semi-supervised setting.
Fig. 7: Semi-supervised setting. Study of linear correlations between the metrics and the accuracy of a LR on the $NQ$ unlabeled samples available during training. In (a), (b), (c) and (d), the data samples come from mini-ImageNet. Their features are extracted with *wideresnet* and diffused through a similarity graph. In (e), (f), (g) and (h), the samples come from tiered-ImageNet. Their features are extracted with *densenet-t* and diffused through a similarity graph. See Section II for details. By default, 5-way 5-shot 30-query tasks are generated. In (a) and (e), the number of queries varies. In (b) and (f), the number of shots varies. In (c) and (g), the number of classes varies. Each point is obtained over 10,000 random tasks. In (d) and (h), each point represents a task. We plot the accuracy of the LR in function of the LR confidence. In (d), 5-way 5-shot 30-query tasks are generated from mini-ImageNet. In (h), 5-way 5-shot 30-query tasks are generated from tiered-ImageNet.

C. Semi-supervised setting

We consider $N$-way $K$-shot $Q$-query tasks. The query samples are available without their labels during training. We study the linear correlation between the LR confidence and the accuracy of the LR on the query samples. In practice, on $NQ$ samples. We also look at the correlations obtained with the metrics defined on supervised and unsupervised inputs. In the unsupervised case, we consider all training samples as unlabeled samples. In Fig. 7 we perform experiments on mini-ImageNet ((a), (b), (c) and (d)) and on tiered-ImageNet ((e), (f), (g) and (h)). In (a) and (e), we consider 5-way 5-shot tasks. The number of queries varies. In (b) and (f), we consider 5-way 30-query tasks. The number of shots varies. In (c) and (g), we consider 5-shot 30-query tasks. The number of classes varies. In (d) and (h), there are two scatter plots. Each point represents a task, with the LR confidence on the X-axis and the accuracy on the Y-axis. In both cases, 5-way 5-shot 30-query tasks are considered. In Appendix A, the average accuracies of the LR for each type of task are given.

**Observations:** We observe that the LR confidence, a metric adapted to the semi-supervised setting, outperforms the supervised and the unsupervised metrics. The supervised metrics seem to depend on the number of unlabeled samples per class $Q$. This is, in part, an artifact due to the fact that the accuracies are computed on the number of queries $NQ$. If $Q$ is small, the range of possible accuracies is reduced, so the computed correlation is affected. The more queries there are, the better the correlation is, with a threshold around 15 additional unlabeled samples per class. The correlation depends less on the number of shots and classes. We observe that the correlation between the supervised metrics and the accuracy are higher than in the supervised setting for the small $K$. This is due to the fact that the features are previously diffused on a similarity graph.

**Discussion:** The LR confidence uses more information than the unsupervised metrics (labels) and the supervised metrics (more data). As in all metrics, a small number of shots does not reduce the linear correlation, we assume that this is due to the diffusion of the features before the training of the LR.

As in the unsupervised setting, we do not have access to the labels of the samples during the training but, we use as many samples per class. In subsection VI-F, we explore the impact of an unbalanced distribution. We also propose an additional experiment on the relevance of different eigenvalues. In subsection VI-F in order to see if we can use the LR confidence to increase the accuracy of the LR on some tasks, we propose to label the data samples with the lowest confidences.

D. Predicting task accuracy

We want to study the relevance of some metrics as proxies to the generalization performance on a set of unseen labeled data samples. In the previous subsections, we look at the
linear correlations between the metrics and the performances on a test set. They measure the joint variation between both variables, but not the ability of the metrics to predict the performances. To go further in this section, we evaluate to what extent the metrics can predict the accuracy of a classifier.

We consider three settings: supervised, unsupervised and semi-supervised. In the supervised setting, we try to predict the accuracy of a LR. In the unsupervised setting and semi-supervised setting, we try to predict the accuracy of a LR when the features have been previously diffused over a similarity graph. To keep the experiments as interpretable as possible, we propose a binary classification problem. The tasks having an accuracy below 80% are hard and the ones above 80% are easy, 80% being an arbitrary choice.

In each setting, we wonder which threshold value of metrics enables to distinguish the best between hard and easy tasks. We plot a ROC curve. The x-coordinate is 1 – specificity. It indicates the proportion of tasks that have been classified as hard tasks among the easy tasks. The y-coordinate is the sensibility. It indicates the proportion of tasks that have been classified as hard tasks among the hard tasks. Knowing these two variables, we could choose an adequate threshold value. Following the notations in Fig. 8, we have:

\[
1 - \text{specificity} = \frac{\text{FP}}{\text{TN} + \text{FP}}, \quad (20)
\]

\[
\text{sensibility} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (21)
\]

In practice, we divide the mini-ImageNet dataset into two sets. Both containing 10 classes. On both sets, 10,000 5-way 5-shot (30-query) tasks are randomly generated. In the unsupervised case, we consider all training samples as unlabeled samples. In Fig. 8, we plot the ROC curve using the first set of tasks. After choosing the threshold value, we report a confusion matrix on the second set. We normalize each row of the confusion matrix, so that its first row indicates the percentage of tasks predicted hard or easy among the hard tasks and its second row indicates the percentage of tasks predicted hard or easy among the easy tasks.

**Supervised setting:** In Fig. 9 (a), the ROC curve is built by varying a threshold value over the LR loss. When selecting a threshold value at (0.30, 0.81), we obtain a confusion matrix on the second set where 1 – specificity becomes 0.64 and sensibility becomes 0.95. Here, both variables are higher. Once again, the chosen threshold does not apply to the second set.

**Unsupervised setting:** In Fig. 9 (b), the ROC curve is built by varying a threshold value over the DB-score. When selecting a threshold value at (0.30, 0.81), we show on the confusion matrix that 1 – specificity becomes 0.64 and sensibility becomes 0.95. Here, both variables are higher. Once again, the chosen threshold does not apply to the second set.

**Semi-supervised setting:** In Fig. 9 (c), the ROC curve is built by varying a threshold value over the LR confidence. We select the threshold value at (0.16, 0.81). In the confusion matrix, 1 – specificity becomes 0.76 and sensibility becomes 0.18. Both variables are similar on the two sets. So, the chosen threshold value generalizes to the second set.

Moreover, we perform a second experiment considering only the LR confidence in a semi-supervised setting. The advantage of the LR confidence over other metrics is that it is easily interpretable. Indeed, it associates with each task

![Fig. 8: Notations](image)

![Fig. 9: Task prediction. The ROC curves are computed over a subset of mini-ImageNet containing 10 classes. The tables are computed on the remaining 10 classes, applying the threshold value denoted by a red point on the curves. Features are extracted with wideresnet. In both cases, 10,000 5-way 5-shot (30-query) are randomly generated. In (a), the variable is the LR loss, in (b), the DB-score, and in (c), the LR confidence.](image)
the average confidence of the LR on each query sample. We propose to consider the LR confidence value as a predicted accuracy. We perform an experiment on 10,000 5-way 5-shot tasks. The average error between the real and the predicted accuracies is 2.40%. To evaluate this result, we also compute the mean absolute deviation of the real accuracies from their average. It amounts to compare the LR confidence with a naive method always predicting the same accuracy. The mean absolute deviation is 4.14%. Thus, the predictions of the LR confidence are better than the naive method.

E. Additional experiments exploring different parameters

In previous experiments, they are as many unlabeled samples per class. We propose to explore what happens when their distribution is unbalanced between classes.

In Fig. 10 first column, we propose an experiment on 2-way 5-shot 50-query tasks, where we vary the proportion of unlabeled data samples in the first class with respect to the second one. In the unsupervised case, we consider that all samples are unlabeled. In both semi-supervised (a) and unsupervised (c) settings, the linear correlations tend to decrease with the imbalance. The supervised metrics (LR loss and similarity) do not take into account the unlabeled samples. However, whereas they are computed on balanced labeled samples, the accuracy is computed on an unbalanced set of samples. This may explain the decrease. The linear correlations with the LR confidence are rather constant. As the LR confidence is directly linked to the query samples and the LR, it is more robust. In the semi-supervised setting, the linear correlations with the DB-score and the N-th eigenvalue go to 0. The DB-score measures the quality of the N clusters made by a N-means on the unlabeled samples. The N-th eigenvalue measures to what extent a 15 nearest neighbors similarity graph computed on the unbalanced data samples is far from having N connected components. When the distribution of the unlabeled samples is unbalanced, these measures on clusters/connected components no longer represent what happens in the LR. In the unsupervised setting, the DB-score stays correlated with the ARI.

In Fig. 10 second column, we propose an experiment on 5-way 5-shot 50-query tasks, where we vary the proportion of unlabeled data samples in a class with respect to the four other classes. The four classes keep the same number of samples. When the number of unlabeled samples in the first class is closed to 0, the problem amounts to a 4-way classification problem with balanced samples, so the correlations are not really influenced. When the number of unlabeled samples in the first class goes to 1, the performances of all metrics, except the LR confidence, decrease. The same reasons as in the 2-way experiment explain the results.

When we motivated the use of the N-th lower eigenvalue as a metric, we assumed that the N classes should correspond to N connected components in a graph whose edges only connect the most similar samples. However, in practice, it is expected that the number of components differ and as such more useful information could be carried by other eigenvalues. In Fig. 11 we report the linear correlation in function of the number of ways where the N-way eigenvalue and the best one are plotted.
Fig. 12: Using per-sample confidence to label data. In a semi-supervised setting, we label some of the unlabeled samples available during training, either randomly or based on the lowest confidences attributed by the LR. At first, we consider 5,000 random 5-way 1-shot 50-query tasks. We recompute the accuracies obtained after labeling some samples. The data come from mini-ImageNet. Their features are extracted with wideresnet and diffused through a similarity graph.

In both semi-supervised and unsupervised settings, the index of the best eigenvalue is lower than the one expected. The performance gap increases with the number of ways.

F. Using per-sample confidence to choose samples to label

Additionally, we propose a last experiment in a semi-supervised setting. Using the LR confidence, we can attribute to each query sample a confidence value. This confidence can be used to decide which unlabeled samples should be labeled for increasing the accuracy of the classifier. In Fig. 12 we compare what happens when labeling specific query examples, and when labeling examples at random. We observe that when the number of labeled samples is small, it is better to have a random selection. This is due to the fact the sampling is more balanced between classes than when using the proposed method. Above a certain amount of labeled samples, it becomes clearly more efficient to choose which samples to label. This is not surprising as the chosen elements happen to be the ones with the lowest confidences, meaning that the remaining ones are easy to classify.

VII. Conclusion

In this paper, we introduced the problem of predicting the generalization performance of few-shot solutions, taking into account the fact they do not have access to a large enough validation set. We first identified that the difficulty of the considered task relies mainly on the relation between base classes, used to train the feature extractor, and novel classes. We then introduced several measures that we showed to be correlated to generalization performance in various settings: supervised, unsupervised and semi-supervised. Interestingly, one of these measures can predict quite well the quality of generalization, despite the lack of labeled data. In future work, we would like to inquire in more details how these findings could help in designing more efficient solutions for the few-shot problem, for example by choosing which samples to label.
