

---

# Leveraging the Feature Distribution in Transfer-based Few-Shot Learning

---

**Yuqing Hu**

Electronics Dept., IMT Atlantique, France - Orange Labs, France  
Cesson-Sévigné  
yuqing.hu@imt-atlantique.fr

**Vincent Gripon**

Electronics Dept., IMT Atlantique, France  
Brest  
vincent.gripon@imt-atlantique.fr

**Stéphane Pateux**

Orange Labs, France  
Cesson-Sévigné  
stephane.pateux@orange.com

## Abstract

Few-shot classification is a challenging problem due to the uncertainty caused by using few labelled samples. In the past few years, transfer-based methods have proved to achieve the best performance, thanks to well-thought-out backbone architectures combined with efficient postprocessing steps. Following this vein, in this paper we propose a transfer-based novel method that builds on two steps: 1) preprocessing the feature vectors so that they become closer to Gaussian-like distributions, and 2) leveraging this preprocessing using an optimal-transport inspired algorithm. Using standardized vision benchmarks, we prove the ability of the proposed methodology to achieve state-of-the-art accuracy with various datasets, backbone architectures and few-shot settings.

## 1 Introduction

Thanks to their outstanding performance, Deep Learning methods are widely considered for vision tasks such as object classification and detection. These systems are typically trained using very large labelled datasets that are representative enough of the inputs to be processed afterwards.

However, in many applications, it is costly to acquire or to annotate data, resulting in the impossibility to create such large labelled datasets. In this context, it is challenging to train Deep Learning architectures considering the fact they typically are made of way more parameters than the dataset contains. This is why in the past few years, few-shot learning (i.e. the problem of learning with few labelled examples) has become a trending research subject in the field.

Most works in the domain are built based on a "learning to learn" guidance, where the pipeline is to train an optimizer [8, 19, 26] with different tasks of limited data so that the model is able to learn generic experience for novel tasks. Namely, the model learns a set of initialization parameters that are in an advantageous position for the model to adapt to a new (small) dataset. Recently, the trend evolved towards using well-thought-out transfer architectures (called backbones) [27, 7], typically trained on large available labelled datasets that differ in their content from the ones of the task.

A main problem of using feature vectors extracted using a backbone architecture is that their distribution is likely to be complex, as the backbone optimized problem most of the time differs from the considered task. As such, methods that rely on strong assumptions about the data distributions are likely to fail in leveraging the quality of features. In this paper, we tackle the problem of transfer-based few-shot learning with a twofold strategy: 1) preprocessing the data extracted from the backbone

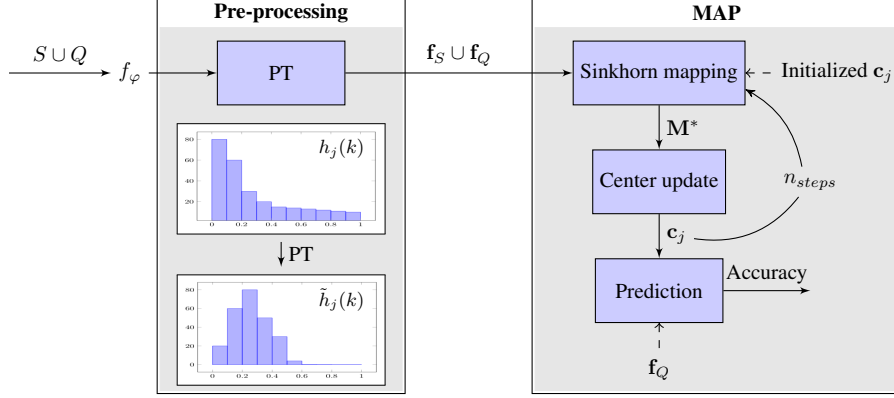


Figure 1: Illustration of the proposed method. First we extract feature vectors of all the inputs in  $\mathbf{D}_{novel}$  and pre-process them to obtain  $\mathbf{f}_S \cup \mathbf{f}_Q$ , note that the Power transform (PT) has the effect of transforming a skewed feature distribution into a gaussian-like distribution ( $h_j(k)$  denotes the histogram of feature  $k$  in class  $j$ ). In MAP, we perform Sinkhorn mapping with class center  $\mathbf{c}_j$  initialized on  $\mathbf{f}_S$  to obtain the class allocation matrix  $\mathbf{M}^*$  for  $\mathbf{f}_Q$ , and we update the class centers for the next iteration. After  $n_{steps}$  we evaluate the accuracy on  $\mathbf{f}_Q$ .

so that it fits a particular distribution (i.e. Gaussian-like) and 2) leveraging this specific distribution thanks to a well-thought proposed algorithm based on maximum a posteriori and optimal transport. Using standardized benchmarks in the field, we demonstrate the ability of the proposed method to obtain state-of-the-art accuracy, for various problems and backbone architectures.

## 2 Related work

A large volume of works in few-shot classification is based on meta learning [26] methods, where the training data is transformed into episodes to better fit in the context of few examples. In this branch, optimization based methods [26, 8, 19] train a well-initialized optimizer so that it quickly adapts to unseen classes with a few epochs of training. Other works [36, 4] utilize data augmentation techniques to artificially increase the size of the training datasets.

In the past few years, there have been a growing interest in transfer-based methods. The main idea consists in training feature extractors able to efficiently segregate novel classes it never saw before. For example, in [3] the authors train the backbone with distance-based classifier [18] that takes into account the inter-class distance. In [17], the authors utilize self-supervised learning techniques [2] to co-train an extra rotation classifier for the output features, improving the accuracy in few-shot settings. Many approaches are built on top of a feature extractor. For instance, in [34] the authors implement a nearest class mean classifier to associate an input with a class whose centroid is the closest in terms of the  $\ell_2$  distance. In [14] an iterative approach is used to adjust class centers. In [11] the authors build a graph neural neural network to gather the neighbor feature information.

Although many works involve feature extraction, few have explored the features in terms of their distribution [9]. Often, assumptions are made that the features in a class align to a certain distribution, even though these assumptions are rarely experimentally discussed. In our work, we analyze the impact of the features distributions and how they can be transformed for better processing and accuracy. We also introduce a new algorithm to improve the quality of the association between input features and corresponding classes in typical few-shot settings.

**Contributions.** Let us highlight the main contributions of this work. (1) We propose to pre-process the raw extracted features in order to make them more aligned with Gaussian assumptions. Namely we introduce transforms of the features so that they become less skewed. (2) We use a wasserstein-based method to better align the distribution of features with that of the considered classes. (3) We show that the proposed method can bring large increase in accuracy with a variety of feature extractors and datasets, leading to state-of-the-art results in the considered benchmarks.

### 3 Methodology

In this section we introduce the problem settings. We discuss the training of the feature extractors, the pre-processing steps that we apply on the trained features and the final classification algorithm. A summary of our proposed method is depicted in Figure 1.

#### 3.1 Problem statement

We consider a typical few-shot learning problem. We are given a *base* dataset  $\mathbf{D}_{base}$  and a *novel* dataset  $\mathbf{D}_{novel}$  such that  $\mathbf{D}_{base} \cap \mathbf{D}_{novel} = \emptyset$ .  $\mathbf{D}_{base}$  contains a large number of labelled examples from  $K$  different classes.  $\mathbf{D}_{novel}$ , also referred as a task in other works, contains a small number of labelled examples (support set  $S$ ), along with some unlabelled ones (query set  $Q$ ), all from  $w$  new classes. Our goal is to predict the class of the unlabelled examples in the query set. The following parameters are of particular importance to define such a few-shot problem: the number of classes in the novel dataset  $w$  (called  $w$ -ways), the number of labelled samples per class  $s$  (called  $s$ -shot) and the number of unlabelled samples per class  $q$ . So the novel dataset contains a total of  $w(s + q)$  samples,  $ws$  of them being labelled, and  $wq$  of them being those to classify.

#### 3.2 Feature extraction

The first step is to train a neural network backbone model using only the base dataset. In this work we consider multiple backbones, with various training procedures. Namely, we use two self-supervised models built upon Wideresnet and Resnet architectures [17], as well as a Densenet backbone [34]. Once the considered backbone is trained, we obtain robust embeddings that should generalize well to novel classes. We denote by  $f_\varphi$  the backbone function, obtained by extracting the output of the penultimate layer from the considered architecture, with  $\varphi$  being the trained architecture parameters. Note that importantly, in all backbone architectures used in the experiments of this work, the penultimate layers are obtained by applying a ReLU function, so that all feature components coming out of  $f_\varphi$  are nonnegative.

#### 3.3 Feature pre-processing

As mentioned in Section 2, many works hypothesize, explicitly or not, that the features from the same class are aligned with a specific distribution (often Gaussian-like). But this aspect is rarely experimentally verified. In fact, it is very likely that features obtained using the backbone architecture are not Gaussian. For example, consider the toy example where a novel class consists of the union of two of the base classes. In this scenario, the backbone has been trained to discriminate between those two base classes where in the novel task we would like the corresponding inputs to be grouped.

Multiple works in the domain [34, 14] discuss the different statistical methods (e.g. normalization) to better fit the features into a model. Although these methods may have provable assets for some distributions, they could worsen the process if applied to an unexpected input distribution. This is why we propose to pre-process the obtained feature vectors so that they better align with typical distribution assumptions in the field. Namely, we use a power transform as follows.

**Power transform (PT).** Denote  $\mathbf{v} = f_\varphi(\mathbf{x}) \in \mathbb{R}^d$ ,  $\mathbf{x} \in \mathbf{D}_{novel}$  as the obtained features on  $\mathbf{D}_{novel}$ . We hereby perform a power transformation method, which is similar to Tukey’s Transformation Ladder [28], on the features. We then follow a unit variance projection, the formula is given by:

$$f(\mathbf{v}) = \begin{cases} \frac{(\mathbf{v}+\epsilon)^\beta}{\|(\mathbf{v}+\epsilon)^\beta\|_2} & \text{if } \beta \neq 0 \\ \frac{\log(\mathbf{v}+\epsilon)}{\|\log(\mathbf{v}+\epsilon)\|_2} & \text{if } \beta = 0 \end{cases}, \quad (1)$$

where  $\epsilon = 1e - 6$  is used to make sure that  $\mathbf{v} + \epsilon$  is strictly positive and  $\beta$  is a hyper-parameter. The rationales of the pre-processing above are: (1) Power transforms have the functionality of reducing the skew of a distribution, adjusted by  $\beta$ , (2) Unit variance projection scales the features to the same area so that large variance feature doesn’t predominate the others. Given that a Gaussian distribution is desirable in many scenarios, this pre-processing step is thus able to map data from any distribution to a close-to-Gaussian distribution.

Note that  $\beta = 1$  leads to almost no effect. More generally, the skew of the obtained distribution changes when  $\beta$  varies. For instance, if a raw distribution is right-skewed, decreasing  $\beta$  phases out the right skew, and phases into a left-skewed distribution when  $\beta$  becomes negative. After experiments, we found that  $\beta = 0.5$  gives the most consistent results for our considered experiments. More details based on our considered experiments are available in Section 4.

### 3.4 MAP

Let us assume that the preprocessed feature distribution for each class is Gaussian or Gaussian-like. As such, a well-positioned class center is crucial to a good prediction. In this section we discuss how to best estimate the class centers when the number of samples is very limited and classes are only partially labelled. In more details, we propose to exploit the prior that we have the same number of samples from each class. Under this assumption we will firstly show that estimating these centers through Maximum A Posteriori (MAP) is similar to the minimisation of Wasserstein distance. Therefore, an iterative procedure based on a Wasserstein distance estimation, using the sinkhorn algorithm [6, 29], is designed to estimate the optimal transport from the initial distribution of the feature vectors to one that would correspond to a balanced draw of samples from Gaussian distributions. Note that here we adopt what is often called a transductive setting in the literature [16, 14], where we exploit the unlabelled samples during the procedure.

In the following, we denote by  $\mathbf{f}_S$  the set of feature vectors corresponding to labelled inputs and by  $\mathbf{f}_Q$  the set of feature vectors corresponding to unlabelled inputs. For a feature vector  $\mathbf{f} \in \mathbf{f}_S \cup \mathbf{f}_Q$ , we denote by  $\ell(\mathbf{f})$  the corresponding label. We use  $0 < i \leq wq$  to denote the index of an unlabelled sample, so that  $\mathbf{f}_Q = (\mathbf{f}_i)_i$ , and we denote  $\mathbf{c}_j, 0 < j \leq w$  as the center of class  $j$ .

Our algorithm consists in several steps in which we estimate class centers from a soft allocation matrix, then we update the allocation matrix based on the newly found class centers. In the following paragraphs, we detail these steps.

**Sinkhorn mapping.** If we consider using MAP estimation for the class centers, assuming a Gaussian distribution for each class, we would typically try to solve:

$$\begin{aligned} \{\hat{\ell}(\mathbf{f}_i)\}, \{\hat{\mathbf{c}}_j\} &= \arg \max_{\{\ell(\mathbf{f}_i)\} \in \mathcal{C}, \{\mathbf{c}_j\}} \prod_i P(\mathbf{f}_i | j = \ell(\mathbf{f}_i)) \\ &= \arg \min_{\{\ell(\mathbf{f}_i)\} \in \mathcal{C}, \{\mathbf{c}_j\}} \sum_i (\mathbf{f}_i - \mathbf{c}_{\ell(\mathbf{f}_i)})^2, \end{aligned} \quad (2)$$

where  $\mathcal{C}$  represents the set of admissible labelling sets. We can observe that this last term corresponds to the Wasserstein distance used in the Optimal Transport problem [6].

Therefore, inspired by the sinkhorn algorithm [31, 6], we define the mapping matrix as follows:

$$\mathbf{M}^* = \text{Sinkhorn}(\mathbf{L}, \mathbf{p}, \mathbf{q}, \lambda) = \arg \min_{\mathbf{M} \in \mathbb{U}(\mathbf{p}, \mathbf{q})} \sum_{ij} \mathbf{M}_{ij} \mathbf{L}_{ij} + \lambda H(\mathbf{M}), \quad (3)$$

where  $\mathbb{U}(\mathbf{p}, \mathbf{q}) \in \mathbb{R}_+^{wq \times w}$  is a set of positive matrices for which the rows sum to  $\mathbf{p}$  and columns sum to  $\mathbf{q}$ . The formula is given by:

$$\mathbb{U}(\mathbf{p}, \mathbf{q}) = \{\mathbf{M} \in \mathbb{R}_+^{wq \times w} | \mathbf{M} \mathbf{1}_w = \mathbf{p}, \mathbf{M}^T \mathbf{1}_{wq} = \mathbf{q}\}, \quad (4)$$

where  $\mathbf{p}$  denotes the distribution of the amount that each unlabelled example uses for class allocation, and  $\mathbf{q}$  denotes the distribution of the amount of unlabelled examples allocated to each class. Therefore,  $\mathbb{U}(\mathbf{p}, \mathbf{q})$  contains all the possible ways of allocating examples to classes. Note that here we assume a soft class mapping, meaning that each example can be 'sliced' into different classes. To keep the consistency of the Gaussian assumption, here the cost function  $\mathbf{L} \in \mathbb{R}^{wq \times w}$  in Equation (3) consists of the euclidean distances between unlabelled examples and class centers, hence  $\mathbf{L}_{ij}$  denotes the euclidean distance between example  $i$  and class center  $j$ .

The second term on the right of Equation (3) denotes the entropy of  $\mathbf{M}$ :  $H(\mathbf{M}) = -\sum_{ij} \mathbf{M}_{ij} \log \mathbf{M}_{ij}$ , regularized by a hyper-parameter  $\lambda$ . Increasing  $\lambda$  would force the entropy to become smaller, so that the mapping is less homogeneous. This term also makes the objective function strictly convex [6, 24] and thus a practical and effective computation.

**Iterative center estimation.** In this step, our aim is to estimate class centers with an increasing accuracy. As shown in Algorithm 1, we initialize  $\mathbf{c}_j$  as the average of labelled samples belonging to

class  $j$ . Then  $\mathbf{c}_j$  is iteratively re-estimated. At each step, we compute a mapping matrix  $\mathbf{M}$  on the unlabelled examples using the sinkhorn mapping. Along with labelled examples, we re-estimate  $\mathbf{c}_j$  (denoted  $\boldsymbol{\mu}_j$ ) by weighted-averaging the features with their allocated portions for class  $j$ :

$$\boldsymbol{\mu}_j = g(\mathbf{M}^*, j) = \frac{\sum_{i=1}^{wq} \mathbf{M}^*_{ij} \mathbf{f}_i + \sum_{\mathbf{f} \in \mathbf{f}_S, \ell(\mathbf{f})=j} \mathbf{f}}{s + \sum_{i=1}^{wq} \mathbf{M}^*_{ij}}. \quad (5)$$

This formula corresponds to the minimization of Equation (3). Note that labelled examples do not participate in the mapping process. Since their labels are known, we instead set allocations for their belonging classes to be 1 and the others to be 0. Therefore, labelled examples have the largest weight when re-estimating the class centers.

**Proportioned center update.** In order to avoid taking risky harsh decisions in early iterations of the algorithm, we propose to proportionate the update of class centers using an inertia parameter. In more details, we update the center with a learning rate  $0 < \alpha \leq 1$ :

$$\mathbf{c}_j \leftarrow \mathbf{c}_j + \alpha(\boldsymbol{\mu}_j - \mathbf{c}_j), \quad (6)$$

so that the center moves at slower pace at each step.

A summary of our proposed algorithm is presented in Algorithm 1. In Table 1 we summarize the main parameters and hyperparameters of the considered problem and proposed solution. The code is available at <https://github.com/yhu01/PT-MAP>.

---

**Algorithm 1:** Proposed algorithm

---

**Parameters :**  $w, s, q, \lambda, \alpha, n_{steps}$

**Initialization :**  $\mathbf{c}_j = \frac{1}{s} \cdot \sum_{\mathbf{f} \in \mathbf{f}_S, \ell(\mathbf{f})=j} \mathbf{f}$

**repeat**  $n_{steps}$  **times:**

$\mathbf{L}_{ij} = \|\mathbf{f}_i - \mathbf{c}_j\|^2, \forall i, j$   
 $\mathbf{M}^* = \text{Sinkhorn}(\mathbf{L}, \mathbf{p} = \mathbf{1}_{wq}, \mathbf{q} = q\mathbf{1}_w, \lambda)$   
 $\boldsymbol{\mu}_j = g(\mathbf{M}^*, j)$   
 $\mathbf{c}_j \leftarrow \mathbf{c}_j + \alpha(\boldsymbol{\mu}_j - \mathbf{c}_j)$

**end**

**return**  $\hat{\ell}(\mathbf{f}_i) = \arg \max_j (\mathbf{M}^*(\ell(\mathbf{f}_i) = j))$

---

Table 1: Important parameters and hyperparameters of our problem.

Novel dataset parameters		
Notation	Value	Description
$w$	typically 5	number of classes
$s$	typically 1 or 5	number of labelled inputs per class
$q$	typically 15	number of unlabelled inputs per class
Proposed method hyperparameters		
Notation	Range	Description
$\beta$	$\{-2, -1, -0.5, 0, 0.5, 1, 2\}$	coefficient to adjust distribution skew
$\lambda$	$\lambda \in \mathbb{R}_+$	regularization coefficient for sinkhorn mapping
$\alpha$	$0 < \alpha \leq 1$	learning rate for class center update

## 4 Experiments

### 4.1 Datasets

We evaluate the performance of the proposed method using standardized few-shot classification datasets: miniImageNet [32], tieredImageNet [20], CUB [33] and CIFAR-FS [1]. The **miniImageNet** dataset contains 100 classes randomly chosen from ILSVRC- 2012 [21] and 600 images of size  $84 \times 84$  pixels per class. It is split into 64 base classes, 16 validation classes and 20 novel classes. The **tieredImageNet** is another subset of ImageNet, the dataset consists of 34 high-level categories with 608 classes in total. These categories are split into 20 meta-training superclasses, 6 meta-validation

superclasses and 8 meta-test superclasses, which corresponds to 391 base classes, 97 validation classes and 160 novel classes respectively. The **CUB** dataset contains 200 classes and has 11,788 images of size  $84 \times 84$  pixels in total. Following [11], it is split into 100 base classes, 50 validation classes and 50 novel classes. The **CIFAR-FS** dataset has 100 classes, each class contains 600 images of size  $32 \times 32$  pixels. The splits of this dataset are the same as those in miniImageNet.

## 4.2 Implementation details

In order to stress the genericity of our proposed method with regards to the chosen backbone architecture and training strategy, we perform experiments using **WRN** [35, 17], **ResNet18** [10, 34] and **DenseNet121** [12, 34] as backbones. For each dataset we train the feature extractor with base classes, tune the hyperparameters with validation classes and test the performance using novel classes. Therefore, for each test run,  $w$  classes are drawn uniformly at random among novel classes. Among these  $w$  classes,  $s$  labelled examples and  $q$  unlabelled examples per class are uniformly drawn at random to form  $\mathbf{D}_{\text{novel}}$ . In our settings, the **WRN** and **ResNet18** are trained following [17], while we train the **DenseNet121** by following [34]. In order to better segregate between feature vectors of corresponding classes for each task, we implement the "trans-mean-sub" [14] before MAP where we separately subtract inputs by the means of labelled and unlabelled examples, followed by a unit hypersphere projection. All our experiments are performed using  $w = 5, q = 15, s = 1$  or  $5$ . We run 10,000 random draws to obtain mean accuracy score and indicate confidence scores (95%) when relevant. The tuned hyperparameters for miniImageNet are  $\beta = 0.5, \lambda = 10, \alpha = 0.4$  and  $n_{\text{steps}} = 30$  for  $s = 1$ ;  $\beta = 0.5, \lambda = 10, \alpha = 0.2$  and  $n_{\text{steps}} = 20$  for  $s = 5$ . Hyperparameters for other datasets are detailed in the experiments below.

## 4.3 Comparison with state-of-the-art methods

In the first experiment, we conduct our proposed method on different benchmark datasets and compare the performance with other state-of-the-art solutions. The results are presented in Table 2, we observe that our method reaches the state-of-the-art performance, and ranks at top #1 for  $s = 1$  and  $s = 5$  on all the benchmarks. Note that for tieredImageNet, our results are obtained with DenseNet121 that is trained using a different strategy. Therefore, we point out that the proposed method can bring an increase of accuracy with a variety of backbones, leading to competitive performance. In terms of execution time, we measured an average of 0.002s per run.

**Performance on cross-domain settings.** We also test our method in a cross-domain setting, where the backbone is trained with the base classes in miniImageNet but tested with the novel classes in CUB dataset. As shown in Table 3, the proposed method gives the best accuracy both in the case of 1-shot and 5-shot.

## 4.4 Other experiments

**Influence of unlabelled data.** Under the transductive settings, we have access to the entire unlabelled samples. Therefore, the total number of query set items  $wq$  has an impact on the performance. To verify that, we evaluate our proposed MAP method with different datasets and backbones. All tests are conducted by fixing  $w = 5$  and varying  $q$  from 2 to 20. The results of this experiment are presented in Figure 2 (1), where we observe that the unlabelled data plays an important role in increasing the prediction accuracy. Interestingly, the accuracy quickly reaches a close-to-asymptotical plateau, emphasizing the ability of the method to soon exploit available information in the task.

**Ablation study.** We then evaluate the importance of the two steps of our proposed method in the accuracy of the obtained solution. In details, we evaluate the impact of Power transform and Sinkhorn mapping separately. We also consider using Euclidean mapping, where each unlabelled example maps to the considered classes with probabilities computed using euclidean distances between feature vectors and class centers [5], to be the alternative of Sinkhorn mapping for the class center estimation. Table 4 shows the comparison results with different backbones and different training strategies.

We point out that our proposed method with Power transform and Sinkhorn mapping reaches the best accuracy. Power transform alone brings a relatively large increase of accuracy on  $s = 1$  and  $s = 5$  when evaluated with Euclidean mapping for WRN backbone. And Sinkhorn mapping outperforms

Table 2: 1-shot and 5-shot accuracy of state-of-the-art methods in the literature, compared with the proposed solution. We present results using WRN, ResNet18 and DenseNet121 as backbones.

Method	Backbone	miniImageNet	
		1-shot	5-shot
Baseline++ [3]	ResNet18	51.87 $\pm$ 0.77%	75.68 $\pm$ 0.63%
MAML [8] <sup>b</sup>	ResNet18	49.61 $\pm$ 0.92%	65.72 $\pm$ 0.77%
ProtoNet [23] <sup>‡</sup>	WRN	62.60 $\pm$ 0.20%	79.97 $\pm$ 0.14%
Matching Networks [32] <sup>‡</sup>	WRN	64.03 $\pm$ 0.20%	76.32 $\pm$ 0.16%
ACC+Amphibian [25]	WRN	64.21 $\pm$ 0.62%	87.75 $\pm$ 0.73%
SimpleShot [34]	DenseNet121	64.29 $\pm$ 0.20%	81.50 $\pm$ 0.14%
S2M2_R [17]	WRN	64.93 $\pm$ 0.18%	83.18 $\pm$ 0.11%
BD-CSPN [15]	WRN	70.31 $\pm$ 0.93%	81.89 $\pm$ 0.60%
Transfer+SGC [11]	WRN	76.47 $\pm$ 0.23%	85.23 $\pm$ 0.13%
TAFSSL [14]	DenseNet121	77.06 $\pm$ 0.26%	84.99 $\pm$ 0.14%
DFMN-MCT [13]	ResNet12	78.30 $\pm$ 0.81%	86.48 $\pm$ 0.42%
PT+MAP(ours)	ResNet18	80.00 $\pm$ 0.27%	86.96 $\pm$ 0.14%
PT+MAP(ours)	WRN	<b>82.92 <math>\pm</math> 0.26%</b>	<b>88.82 <math>\pm</math> 0.13%</b>
Method	Backbone	tieredImageNet	
		1-shot	5-shot
ProtoNet [23] <sup>‡</sup>	ConvNet4	53.31 $\pm$ 0.89%	72.69 $\pm$ 0.74%
LEO [22] <sup>‡</sup>	WRN	66.33 $\pm$ 0.05%	81.44 $\pm$ 0.09%
SimpleShot [34]	DenseNet121	71.32 $\pm$ 0.22%	86.66 $\pm$ 0.15%
DFMN-MCT [13]	ResNet12	80.89 $\pm$ 0.84%	87.30 $\pm$ 0.49%
TAFSSL [14]	DenseNet121	84.29 $\pm$ 0.25%	89.31 $\pm$ 0.15%
PT+MAP(ours)	DenseNet121	<b>85.41 <math>\pm</math> 0.25%</b>	<b>90.44 <math>\pm</math> 0.14%</b>
Method	Backbone	CUB	
		1-shot	5-shot
Baseline++ [3]	ResNet10	69.55 $\pm$ 0.89%	85.17 $\pm$ 0.50%
MAML [8] <sup>b</sup>	ResNet10	70.32 $\pm$ 0.99%	80.93 $\pm$ 0.71%
ProtoNet [23] <sup>b</sup>	ResNet18	72.99 $\pm$ 0.88%	86.64 $\pm$ 0.51%
Matching Networks [32] <sup>b</sup>	ResNet18	73.49 $\pm$ 0.89%	84.45 $\pm$ 0.58%
S2M2_R [17]	WRN	80.68 $\pm$ 0.81%	90.85 $\pm$ 0.44%
Transfer+SGC [11]	WRN	88.35 $\pm$ 0.19%	92.14 $\pm$ 0.10%
PT+MAP(ours)	WRN	<b>91.55 <math>\pm</math> 0.19%</b>	<b>93.99 <math>\pm</math> 0.10%</b>
Method	Backbone	CIFAR-FS	
		1-shot	5-shot
ProtoNet [23]	ConvNet64	55.50 $\pm$ 0.70%	72.00 $\pm$ 0.60%
MAML [8]	ConvNet32	58.90 $\pm$ 1.90%	71.50 $\pm$ 1.00%
BD-CSPN [15]	WRN	72.13 $\pm$ 1.01%	82.28 $\pm$ 0.69%
ACC+Amphibian [25]	WRN	73.10 $\pm$ 0.50%	89.30 $\pm$ 0.90%
S2M2_R [17]	WRN	74.81 $\pm$ 0.19%	87.47 $\pm$ 0.13%
Transfer+SGC [11]	WRN	83.90 $\pm$ 0.22%	88.76 $\pm$ 0.15%
PT+MAP(ours)	WRN	<b>87.69 <math>\pm</math> 0.23%</b>	<b>90.68 <math>\pm</math> 0.15%</b>

<sup>b</sup>: Results reported in [3].

<sup>‡</sup>: Results reported in [34].

Euclidean mapping along with Power transform. Note that the increase in performance is valid for various backbones trained using various techniques.

**Hyperparameter tuning.** In the next experiment we tune  $\beta$ ,  $\lambda$  and  $\alpha$  on the validation classes of each dataset, and then apply them to test our model on novel classes. We vary each hyperparameter in a certain range and observe the evolution of accuracy to choose the peak that corresponds to the highest prediction. For example, the evolving curve for  $\beta$ ,  $\lambda$  and  $\alpha$  with miniImageNet are presented in Figure 2 (2) to (4). For comparison purpose, we also trace the corresponding curves on novel classes. We draw a dash line on the hyperparameter values where the accuracy on the validation classes peaks, meaning that this is the chosen value resulting in Table 2.

The following observations can be drawn from this experiment: (1) The evolving curves on validation classes (red) and novel classes (blue) have generally similar trend for each hyperparameter. In

Table 3: 1-shot and 5-shot accuracy of state-of-the-art methods when performing cross-domain classification (backbone: WRN).

Method	1-shot	5-shot
Baseline++ [3] <sup>‡</sup>	40.44 $\pm$ 0.75%	56.64 $\pm$ 0.72%
Manifold Mixup [30] <sup>‡</sup>	46.21 $\pm$ 0.77%	66.03 $\pm$ 0.71%
S2M2_R [17]	48.24 $\pm$ 0.84%	70.44 $\pm$ 0.75%
Transfer+SGC [11]	58.63 $\pm$ 0.25%	73.46 $\pm$ 0.17%
PT+MAP(ours)	<b>62.49 <math>\pm</math> 0.32%</b>	<b>76.51 <math>\pm</math> 0.18%</b>

<sup>‡</sup>: Results reported in [17].

Table 4: Comparisons of different methods with different backbones on miniImageNet. Note that we train WRN following [17] while DenseNet121 is trained by following [34].

PT	Euclidean mapping	Sinkhorn mapping	WRN		DenseNet121	
			1-shot	5-shot	1-shot	5-shot
	✓		75.34 $\pm$ 0.27%	84.29 $\pm$ 0.15%	76.93 $\pm$ 0.26%	85.60 $\pm$ 0.14%
		✓	75.60 $\pm$ 0.29%	84.13 $\pm$ 0.16%	78.57 $\pm$ 0.28%	86.21 $\pm$ 0.14%
✓	✓		81.22 $\pm$ 0.24%	88.32 $\pm$ 0.13%	78.26 $\pm$ 0.25%	86.61 $\pm$ 0.13%
✓		✓	<b>82.92 <math>\pm</math> 0.26%</b>	<b>88.82 <math>\pm</math> 0.13%</b>	<b>79.98 <math>\pm</math> 0.28%</b>	<b>87.19 <math>\pm</math> 0.13%</b>

particular, two curves peak at the same  $\beta$  ( $\beta = 0.5$ ) and  $\lambda$  ( $\lambda = 10$ ), meaning that validation classes and novel classes share the same  $\beta$  and  $\lambda$  that reach the highest accuracy. (2) A small  $\lambda$  tends to lead to a homogeneous class partition for  $\mathbf{M}^*$ , where each sample are uniformly allocated to  $w$  classes. Hence the sharp drop on the accuracy when  $\lambda < 5$ . (3) A too small  $\alpha$  results in an insufficient class center update. On the contrary, the impact on a large  $\alpha$  is relatively mild. Overall, it is interesting to point out the little sensitivity of the proposed method accuracy with regards to hyperparameter tuning.

We followed this procedure to find the tuned hyperparameters for each dataset. Therefore, we obtained that working with CUB leads to the the same hyperparameters as miniImageNet. For tieredImageNet and CIFAR-FS, the best accuracy are obtained on validation classes when  $\beta = 0.5$ ,  $\lambda = 10$ ,  $\alpha = 0.3$  for  $s = 1$ ;  $\beta = 0.5$ ,  $\lambda = 10$ ,  $\alpha = 0.2$  for  $s = 5$ .

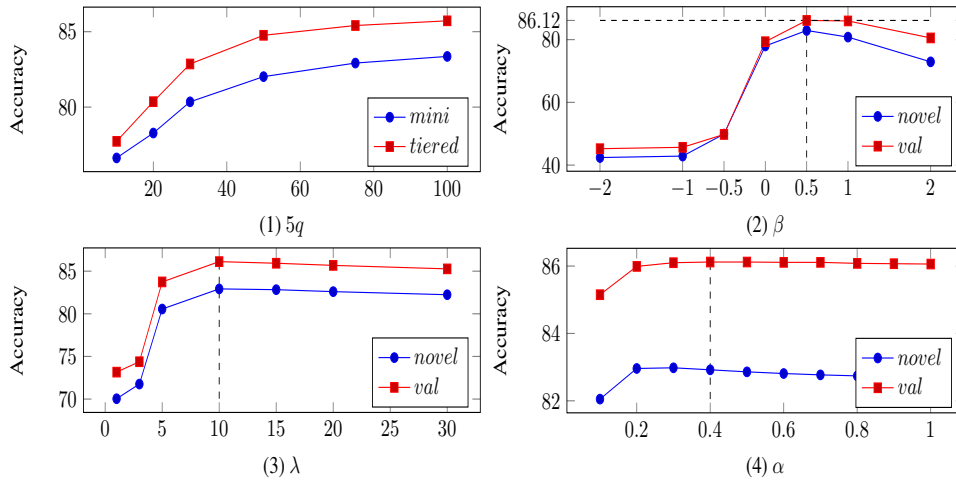


Figure 2: (1) represents 1-shot accuracy on miniImageNet (backbone: WRN) and tieredImageNet (backbone: DenseNet121) as a function of  $q$ . (2), (3) and (4) represent 1-shot accuracy on miniImageNet (backbone: WRN) as a function of  $\beta$ ,  $\lambda$  and  $\alpha$  respectively.



## 5 Conclusion

In this paper we introduced a new pipeline to solve the few-shot classification problem. Namely, we proposed to firstly pre-process the raw feature vectors to better align to a Gaussian distribution and then we designed an optimal-transport inspired iterative algorithm to estimate the class centers. Our experimental results on standard vision benchmarks reach state-of-the-art accuracy, with important gains in both 1-shot and 5-shot classification. Moreover, the proposed method can bring gains with a variety of feature extractors, regardless of their training strategies, with few hyperparameters. Thus we believe that the proposed method is applicable to many practical problems.

## Broader Impact

The main impact of this work is a clear justification of the clustering methodology used in the context of few-shot learning, along with underlying hypothesis made (that are not generally not explicitly given). As such, this work could help in designing few-shot learning systems that are more robust and mathematically sound. There is no specific application targeted.

## Acknowledgments and Disclosure of Funding

This work was partially funded by Orange Labs, Cesson-Sévigné, France.

## References

- [1] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.
- [2] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.
- [3] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification, 2019.
- [4] Z. Chen, Y. Fu, Y.-X. Wang, L. Ma, W. Liu, and M. Hebert. Image deformation meta-networks for one-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8680–8689, 2019.
- [5] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.
- [6] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.
- [7] D. Das and C. G. Lee. A two-stage approach to few-shot learning for image recognition. *IEEE Transactions on Image Processing*, 2019.
- [8] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.
- [9] V. Gripon, G. B. Hacene, M. Löwe, and F. Vermet. Improving accuracy of nonparametric transfer learning via vector segmentation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2966–2970, 2018.
- [10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [11] Y. Hu, V. Gripon, and S. Pateux. Exploiting unsupervised inputs for accurate few-shot classification. *arXiv preprint arXiv:2001.09849*, 2020.
- [12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- [13] S. M. Kye, H. B. Lee, H. Kim, and S. J. Hwang. Transductive few-shot learning with meta-learned confidence. *arXiv preprint arXiv:2002.12017*, 2020.
- [14] M. Lichtenstein, P. Sattigeri, R. Feris, R. Giryes, and L. Karlinsky. Tafssl: Task-adaptive feature sub-space learning for few-shot classification. *arXiv preprint arXiv:2003.06670*, 2020.

- [15] J. Liu, L. Song, and Y. Qin. Prototype rectification for few-shot learning. *arXiv preprint arXiv:1911.10713*, 2019.
- [16] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.
- [17] P. Mangla, N. Kumari, A. Sinha, M. Singh, B. Krishnamurthy, and V. N. Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2218–2227, 2020.
- [18] T. Mensink, J. Verbeek, F. Perronnin, and G. Csorka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision*, pages 488–501. Springer, 2012.
- [19] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2016.
- [20] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.
- [21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- [22] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.
- [23] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.
- [24] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015.
- [25] L. Song, J. Liu, and Y. Qin. Fast and generalized adaptation for few-shot learning. *arXiv preprint arXiv:1911.10807*, 2019.
- [26] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.
- [27] L. Torrey and J. Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.
- [28] J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.
- [29] S. Vallender. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786, 1974.
- [30] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, A. Courville, D. Lopez-Paz, and Y. Bengio. Manifold mixup: Better representations by interpolating hidden states. *arXiv preprint arXiv:1806.05236*, 2018.
- [31] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.
- [32] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.
- [33] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.
- [34] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.
- [35] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
- [36] H. Zhang, J. Zhang, and P. Koniusz. Few-shot learning via saliency-guided hallucination of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2770–2779, 2019.