Revisiting Feature Normalization and Augmentation in Few-shot Learning: a Simple Comparison of Baselines

Anonymous Author(s)
Affiliation
Address
email

Abstract

The growing complexity of network designs makes Few-Shot Learning (FSL) algorithms difficult to compare in a fair manner. Typical FSL methods consist of two parts: a feature extractor trained on base classes, and a predictor tested on a given support set task. Most existing research aim to improve the feature extractor for better generalization, whereas a small number of papers note the predictor design is also important. In this paper, we investigate the predictor module which relies on three components: feature normalization (and transformation), feature augmentation, and a classifier. In comparative ablation experiments we show that (i) with appropriate feature normalization, logistic regression and SVM perform the best in most cases rather than the cosine or nearest neighbour classifier; (ii) feature normalization is very important and the feature augmentation is very helpful in one-shot learning; and (iii) our modified baseline methods (a good selection of existing components) achieve competitive performance when compared with the state of the art on mini-ImageNet, tiered-ImageNet, and the CUB datasets.

1 Introduction

Although deep learning has driven the rapid development of artificial intelligence, with impressive achievements in a wide range of research areas such as vision, text, and speech [9], it is well-known that the success of deep learning relies on a large number of labeled samples. Due to a number of factors such as privacy, security, or high labeling cost of data, many real-world applications (*e.g.*, medical, security, and financial) do not have an access to sufficient labeled training samples. Thus, it is very important to enable deep learning to efficiently learn and generalize from a small number of samples.

Existing Few-Shot Learning (FSL) methods propose different solutions to deal with the low numbers of samples. FSL algorithms can be broadly divided into three categories: metric learning, meta learning, and transfer learning. The goal of a metric-based learning approach is to learn a mapping from images to an embedding space in which images of the same class are pulled closer to each other, whereas images of different classes are pushed apart. We expect metric learnt during the training step to hold for classes that have not been seen before (testing stage). The meta-learning approaches build on adapting to specific tasks from the general set of parameters, and then updating the general set of parameters w.r.t. these tasks. Transfer learning, includes pre-training a feature extractor in the first stage, and then, in the second stage, adapting to reuse this knowledge to obtain a classifier on new samples.

Due to the growing complexity of network designs, FSL algorithms are difficult to compare fairly. Many of the recent advances in FSL use the meta-learning framework, and perform adaptation in the test stage. However, in a recent study [2], it was shown that learning a cosine distance classifier on features extracted from deeper networks also performs quite well on few-shot tasks. Thus sometimes,

Submitted to the 5th Workshop on Meta-Learning at NeurIPS 2021, Sydney, Australia. Do not distribute.

it is difficult to appreciate what components (and what their combination) make FSL work well. In this paper, we focus on a simple perspective on FSL approaches, which includes two steps: training a feature extractor on base classes, and adapting predictor to novel classes. Most existing papers focus on designing a better feature extractor, while only a few papers note that getting a better predictor is equally important. Simple-shot [21] uses a feature normalization before feeding the features into the nearest neighbor classifier, which helps a feature extractor obtain the state-of-the-art performance. Augmentation by adding a noise from the Normal distribution without knowing the prior covariance has also been shown to work well in FSL [3]. Free-lunch FSL,[22] employs a prior computed from the features of base classes. However, there is no systematic discussion or evaluations of possible other strategies of forming the prior.

Thus, in this paper, we break the prediction stage down into its constituent parts, namely, feature normalization, feature augmentation, and classification, as shown in Figure 1. We note that most existing techniques roughly follow such a design. Based on the steps in Figure 1, we conduct a comparative analysis by ablation studies, and we show that:

i. with an appropriate feature normalization, linear classifiers (logistic regression and SVM) perform the best in most cases;

ii. the feature normalization is very important, and the feature augmentation is very helpful for one-shot learning; and

iii. our modified baselines achieve a competitive performance when compared with the state of the art on mini-ImageNet, tiered-ImageNet, and the CUB datasets.

2 Related Work

Feature Augmentation and Normalization. Pixel-level data augmentation techniques have been widely adopted in visual recognition models [7]. They are generic pipelines for augmenting training data with an image-level information. However, feature-wise augmentation (not to be confused with the feature generation [6]) has received the lesser level of attention. A few of pioneering works propose generative feature augmentation strategy for domain adaptation [20], imbalanced classification [25], and FSL [3, 22]. Multi-level semantic feature augmentation [3] employs a feature-level augmentation for one-shot learning by adding noises drawn from a Normal distribution. Because of no prior modeling on that noise, each feature dimension is augmented by drawing from a univariate Normal distribution. In contrast, approaches [3, 22] use the mean and the variance of the distribution evaluated from features of similar classes. The ℓ_2 normalization has been widely used in FSL [2, 21]. Turkey transformation is used in [22] to make the feature distribution closer to the Normal distribution.

Classifiers in Few-Shot Learning. Given limited samples, we restrict the hypothesis space to only simple models (such as linear classifiers). Dynamic few-shot learning [5] learns a weight generator to predict the novel class classifier using an attention-based mechanism (cosine similarity), and Low-shot learning with imprinted weights approach [13] uses novel class features to imprint weights into the classifier. Approach [2] adds an ℓ_2 norm constraint on weights, and uses a softmax over the cosine measure. However, many methods accept taking the nearest neighbor classifier as a solution because it is free of parameters, and thus is able to prevent overfitting, especially in one-shot learning.

FSL based on Transfer Learning. Although approaches [2, 21] pay attention to the classifiers and feature normalization, they mainly advocate that the meta-learning is unnecessary in FSL, and they highlight the importance of the feature extractor. In contrast, we focus on comparative ablations to improve the performance of FSL no matter the backbone or its training regime (meta-learning, similarity learning, *etc*.)

3 Methodology

Below, we define FSL and detail techniques which we use in our comparisons.

3.1 Few-shot Learning

We follow a typical FSL setting. Given a dataset with data-label pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector of a sample, and $y_i \in \mathcal{C}$, where $\mathcal{C}$ denotes the set of classes divided into base

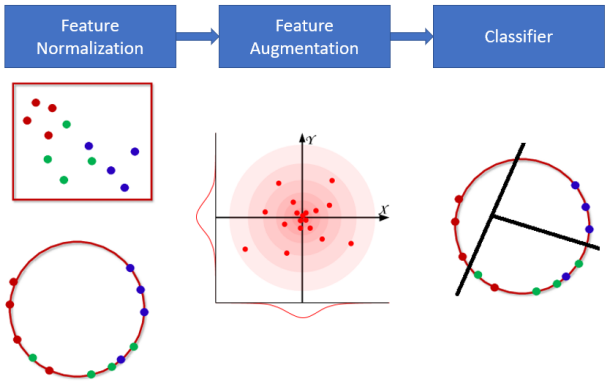

Figure 1: FSL predictor uses the feature normalization, feature augmentation and the classifier.

classes $\mathcal{C}_b$ and novel classes $\mathcal{C}_n$, where $\mathcal{C}_b \cap \mathcal{C}_n = \emptyset$ and $\mathcal{C}_b \cup \mathcal{C}_n = \mathcal{C}$. The goal is to train a feature extractor f_θ on the data from the base classes so that the model can generalize well to the tasks sampled from the novel classes. In order to evaluate the fast adaptation ability or the generalization ability of the model, there are only few labeled samples available for each task T . The most common way to build tasks is the so-called N-way-K-shot protocol [19], where N classes are sampled from the novel set and only K (*e.g.*, 1 or 5) labeled samples are provided for each class. The few available labeled samples form the so-called support set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N \times K}$, whereas the model is evaluated on the query set $Q = \{(\mathbf{x}_i, y_i)\}_{i=N \times K+1}^{N \times K+N \times q}$, with q queries in total. Thus, the performance of a model is evaluated as the average accuracy on the query set containing multiple tasks sampled from the novel classes.

3.2 Feature Normalization

The ℓ_1 and ℓ_2 normalizations. The ℓ_p normalization performs a reprojection of the feature vector on the feasible space induced by the norm (*e.g.*, the simplex or unit ball for the ℓ_1 and ℓ_2 norms, respectively). This is achieved by dividing feature vectors by their ℓ_p norm:

$$g_p(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|_p}, \text{ where } \|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + ... + |x_n|^p)^{1/p}, \tag{1}$$

where x_i is the i-th element of the vector $\mathbf{x}$. In [21], authors proposed a **centered** ℓ_2 normalization, by subtracting the mean of features from base classes and applying the ℓ_2 normalization.

Power Normalization. Power Normalizations (PN) are very useful nonlinear operators as they tackle problems such as feature imbalance [8]. SoSN model [24] uses PN over a co-occurrence matrix (their feature for the deep learner). In this paper, the element-wise square root is considered as a candidate operator, similarly to approach [22]. We also consider a related RootSIFT [1], called **Root** ℓ_1, which uses the ℓ_1 normalization followed by the square root on each individual feature (*i.e.*, $\sqrt{g_1(\mathbf{x})}$). Due to interactions of the ℓ_1 norm with the square root, **Root** ℓ_1 in fact projects the root normalized features on the unit ball rather than the simplex, and it works on non-negative features.

3.3 Feature Vector Augmentation

Below, we introduce three different feature vector augmentation methods, two of which have been proposed in [3] and [22]. The third technique is proposed by us to combine the advantage of the other two.

Diagonal Augmentation (DA) [3]. In this method, the instance features are offset by adding the noise term drawn from the Normal distribution centered at $\mathbf{x}_i$, Thus, we sample the k-th semantic vector $\mathbf{v}_i^k$ as:

$$\mathbf{v}_i^k \sim \mathcal{N}\left(\mathbf{x}_i, \sigma \mathbf{I}\right), \tag{2}$$

where $\sigma \in \mathbb{R}$ is the global variance and $\mathbf{I}$ is the identity matrix, whereas σ controls the standard deviation of the Normal distribution.

Distribution Calibration (DC) [22]. DC extends nearest vectors to k-nearest base classes w.r.t. the feature vector $\mathbf{x}_j$, where the k-th semantic vector $\mathbf{v}_i^k$ is drawn from $\mathbf{x}_i$ as:

$$\mathbf{v}_i^k \sim \mathcal{N}\left(\boldsymbol{\mu}', \boldsymbol{\Sigma}'\right), \text{ where } \boldsymbol{\mu}' = \frac{\sum_{i' \in \mathcal{S}_N} \boldsymbol{\mu}_{i'} + \mathbf{x}_i}{k+1} \text{ and } \boldsymbol{\Sigma}' = \frac{\sum_{i' \in \mathcal{S}_N} \boldsymbol{\Sigma}_{i'}}{k} + \alpha, \tag{3}$$

where $\mathcal{S}_N$ stores the k-nearest base classes with respect to the feature vector $\mathbf{x}_i$.

Prior-based Augmentation (PA). Empirically, we found DC cannot work well with the ℓ_2 normalized features while DA does not leverage the information from base classes. Thus, we propose a simple trade-off augmentation step. We use the mean of the diagonal of covariance matrices from base classes instead of using the isotropic covariance matrix or combining covariance matrices from the k-nearest neighbors of base classes. Specifically, we design a so-called covariance shrinkage [10]. To this end, we sample the k-th semantic vector $\mathbf{v}_i^k$ from the Normal distribution cantered at $\mathbf{x}_i$ as:

$$\mathbf{v}_i^k \sim \mathcal{N}\left(\mathbf{x}_i, \boldsymbol{\Sigma}\right), \text{where } \boldsymbol{\Sigma} = \frac{\sum_{i \in \mathcal{S}_b} \text{diag}\left((1-\epsilon)\boldsymbol{\Sigma}_i + \epsilon \mathbf{I}\right)}{|C_b|}, \tag{4}$$

where $\mathcal{S}_N$ stores all base classes, $0 \leq \epsilon \leq 1$ controls the shrinkage of covariance towards $\mathbf{I}$ (Ledoit and Wolf [10]) to deal with a poor estimate of covariance due to the low-sample regime, diag$(\cdot)$ returns the diagonal of matrix, and $|C_b|$ is the number of base classes.

3.4 Classifier

In this section, we review four simple classifiers used in our comparisons.

Nearest Neighbour Classifier (NNC). Prototypical Networks compute a prototype representation $\bar{\mathbf{x}}_c \in \mathbb{R}^d$ of each class. Each prototype is the mean vector of the support feature vectors belonging to its class c:

$$\bar{\mathbf{x}}_c = \frac{1}{|S_c|} \sum_{(\mathbf{x}, y) \in S_c} \mathbf{x}. \tag{5}$$

Given a distance function $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, prototypical networks produce a distribution over classes for a query point $\mathbf{x}$ based on a softmax over distances to the prototypes:

$$p(y = c \mid \mathbf{x}) = \frac{\exp\left(-\delta^2\left(\mathbf{x}, \bar{\mathbf{x}}_c\right)\right)}{\sum_{c'} \exp\left(-\delta^2\left(\mathbf{x}, \bar{\mathbf{x}}_{c'}\right)\right)}. \tag{6}$$

For the distance function δ we use the ℓ_2 norm. The NNC is a parameter-free version of cosine distance classifier (CDC) [21].

Support Vector Machine (SVM). Unlike the NN classifier in Eq 6, a linear SVM requires learning parameters $\mathbf{w}_c$ for each class c to make predictions on unknown samples. The decision function is given as:

$$\text{argmax}_c \ \mathbf{x}^\top \mathbf{w}_c + b_c, \tag{7}$$

where b_c is the bias for the c-th class prediction.

Multinomial logistic regression (Softmax). Multinomial logistic regression also learns parameters from the training set, and uses the softmax function in the place of the max operator:

$$p(y = c \mid \mathbf{x}) = \frac{\exp\left(\mathbf{x}^\top \mathbf{w}_c + b_c\right)}{\sum_{c'} \exp\left(\mathbf{x}^\top \mathbf{w}_{c'} + b_{c'}\right)}. \tag{8}$$

Cosine Distance Classifier (CDC). CDC is the softmax classifier where feature vectors and weight vectors are ℓ_2-norm normalized, and the bias terms are removed:

$$p(y = c \mid \mathbf{x}) = \frac{\exp\left(\mathbf{x}^\top \mathbf{w}_c / \left(\|\mathbf{x}\|_2 \cdot \|\mathbf{w}_c\|_2\right)\right)}{\sum_{c'} \exp\left(\mathbf{x}^\top \mathbf{w}_{c'} / \left(\|\mathbf{x}\|_2 \cdot \|\mathbf{w}_{c'}\|_2\right)\right)}. \tag{9}$$

4 Experiments

In this section, we answer the following questions: (i) how do different feature normalizations (and transformations) contribute to few-shot classification on different benchmarks; (ii) does feature augmentation help few-shot learning; (iii) which classifier is the best choice for FSL; and (iv) can these techniques yield the state-of-the-arts performance?

Table 1: Comparison of test accuracy against state-of-the art methods for 1-shot and 5-shot classification using mini-ImageNet and tiered-ImageNet.

		mini-ImageNet		tiered-ImageNet	
Methods	Network	1-shot	5-shot	1-shot	5-shot
LEO [17]	WRN	61.76±0.08	77.59±0.12	66.33±0.05	81.44±0.09
ProtoNet [18]	WRN	62.60±0.20	79.97±0.14	–	–
CC+rot [4]	WRN	62.93±0.45	79.87±0.33	70.53±0.51	84.98±0.36
MatchingNet [19]	WRN	64.03±0.20	76.32±0.16	–	–
S2M2 [11]	WRN	64.93±0.18	83.18±0.11	–	–
FEAT [23]	WRN	65.10±0.20	81.11±0.14	70.41±0.23	84.38±0.16
SimpleShot [21]	WRN	65.87±0.20	82.09±0.14	70.90±0.22	85.76±0.15
DC [22]	WRN	68.57±0.55	82.88±0.42	70.42±0.22	87.24±0.14
ℓ_2+SVM	WRN	66.54±0.20	83.51±0.13	69.43±0.22	85.27±0.15
ℓ_2+SVM+PA	WRN	68.02±0.20	82.01±0.11	69.92±0.20	87.02±0.15

Datasets. We experiment on three popular benchmarks. The **mini-ImageNet** dataset [19] is a subset of ImageNet [16] that is commonly used to study few-shot learning. The dataset contains 100 classes and has a total of 600 examples per class. Following [14] and subsequent works, we split it into 64 base, 16 validation, and 20 novel classes. Following [19] and subsequent studies, we resize images to 84×84 pixels and center crop. We also perform experiments on the **tiered-ImageNet** dataset [15], which is also constructed from ImageNet but contains 608 classes. The dataset is split into 351, 97, and 160 classes for base, validation, and novel classes, respectively. The class split is performed using WordNet [12] to ensure that all the base classes are semantically unrelated to the novel classes, and images are resized to 84×84 pixels.

Metrics. We use the top-1 accuracy as the evaluation metric to measure the performance of our method. We report the accuracy on 5-way 1-shot and 5-way 5-shot protocols for mini-ImageNet, tiered-ImageNet and CUB. The reported results are the averaged classification accuracy over 10,000 tasks, unless specified otherwise.

Implementation. Unless specified otherwise, the feature extractor we use is the WideResNet-28-10 (WRN) following the previous work [11]. For each dataset, the feature extractor is trained with base classes and the performance tested on novel classes. Note that the feature representations are extracted from the penultimate layer (with a ReLU activation function) to meet requirements of DC [22] and the Root ℓ_1 normalization.

4.1 Impact of Feature Normalization

Table 2 is the comparison among four different feature normalization methods. In six different settings for three classifiers, the ℓ_2 normalization yields 3 best results, the Root ℓ_1 normalization yields 2 best results and the Centering ℓ_2 normalization yields 1 best result. It is worth noting that although the Rootℓ_1 normalization uses Power Normalization (element-wise square root) and it outperforms the baseline of ℓ_1 normalization, it is hard to conclude if the square root plays an important role because the Root ℓ_1 normalization is not significantly better than the ℓ_2 normalization in most cases, and both methods normalize the features to the unit sphere. In what follows, the normalization we use is set to the ℓ_2 normalization, unless specified otherwise.

4.2 Impact of Feature Augmentation

Below, we compare three different feature augmentation strategies, DA, DC and PA, in 1-shot setting. Empirically, we observe that feature augmentations give better results after feature normalization, so we apply the ℓ_2 normalization. Note that the DC [22] cannot work well with normalized features thus we use the default setting from [22]. Table 3 shows that the feature augmentation plays an important role on mini-ImageNet and tiered-ImageNet but all feature augmentations fail on CUB. This indicates that unlike the feature normalization, the simplistic feature augmentation does not always play a positive role. Therefore, one should be careful in using feature augmentations in practice, if there is a larger number of training samples available.

Table 2: Average accuracy (in %; measured over 10,000 rounds) of 1-shot and 5-shot classifiers for 5-way classification on mini-ImageNet, CUB, tiered-ImageNet; higher % is better. The best result of each combination of each column is in bold font.

Benchmark	Method	ℓ_1	ℓ_2	Root ℓ_1	Centering ℓ_2
mini-ImageNet 1-shot	LR	65.24±0.20	**66.64±0.20**	66.07±0.19	64.21±0.20
	SVM	65.65±0.20	66.54±0.20	66.02±0.19	64.41±0.20
	NNC	65.55±0.20	65.36±0.20	63.11±0.20	63.79±0.20
	CDC	-	64.93±0.18	63.24±0.20	-
mini-ImageNet 5-shot	LR	81.55±0.14	83.50±0.12	83.51±0.13	81.63±0.14
	SVM	81.60±0.14	83.51±0.13	**84.01±0.13**	82.29±0.14
	NNC	78.16±0.16	79.15±0.16	76.27±0.17	77.89±0.16
	CDC	-	83.18±0.11	83.36±0.14	-
CUB 1-shot	LR	79.72±0.20	80.91±0.20	79.37±0.20	80.62±0.20
	SVM	80.01±0.20	80.91±0.20	79.55±0.20	80.73±0.20
	NNC	79.63±0.20	**81.06±0.20**	79.05±0.20	81.00±0.20
	CDC	-	80.68±0.20	79.16±0.16	-
CUB 5-shot	LR	90.19±0.11	91.26±0.10	91.31±0.10	91.23±0.10
	SVM	90.25±0.11	91.26±0.10	**91.50**±0.10	91.18±0.10
	NNC	88.64±0.13	90.44±0.11	90.10±0.12	90.07±0.12
	CDC	-	90.85±0.11	91.24±0.11	-
tiered-ImageNet 1-shot	LR	66.72±0.22	69.40±0.22	69.11±0.21	69.02±0.20
	SVM	67.44±0.22	**69.43±0.22**	69.13±0.20	68.88±0.20
	NNC	64.70±0.22	68.81±0.22	66.45±0.22	68.85±0.21
	CDC	-	68.51±0.22	66.57±0.20	-
tiered-ImageNet 5-shot	LR	82.62±0.16	85.10±0.15	85.34±0.15	85.88±0.14
	SVM	82.76±0.16	85.27±0.15	85.73±0.15	**86.09**±0.14
	NNC	77.43±0.20	82.27±0.19	80.01±0.18	82.01±0.15
	CDC	-	85.53±0.15	66.57±0.20	-

Table 3: Ablation study on 5-way 1-shot protocol on three different datasets showing accuracy (%) with 95% confidence intervals.

	mini-ImageNet	CUB	tiered-ImageNet
S2M2 [11]	65.87±0.20	81.08±0.20	69.12±0.22
DA	67.10±0.20	80.04±0.20	69.43±0.22
DC	68.57±0.20	79.56±0.87	70.42±0.21
PA	68.02±0.17	81.06±0.20	69.92±0.20

4.3 Choice of Classifier

We compare the NNC with the logistic regression, SVM, and CDC. Table 2 shows the comparison among four different classifiers with six different settings and four feature normalization methods. As it turns out, NNC and CDC do not perform better than SVM *e.g.*, NNC yields the best result on the 1-shot tasks on CUB with the ℓ_2 normalization. In fact, given different feature normalization methods, NNC appears as the worst classifier (up to 5% drop) in the comparison, likely because it cannot be tuned on test support samples.

4.4 Comparisons on Different Backbones and Learners

To support our findings in a more broad setting, we compare the results of two learners, SimpleShot [21] and S2M2 [11], given the same backbone (WideResNet-28-10), as shown in Table 4. We further compare the performance of ResNet-10,18,34,50, MobileNet and WRN on mini-ImageNet and tiered-ImageNet given 5-way 1-shot setting, as shown in Table 5 and 6. We observe that, compared with baselines [21] (ℓ_2 normalization + nearest neighbour classifier), SVM and the feature augmentation do improve the performance. We do not evaluate ResNet-50 on tiered-Imagenet because the ResNet-50 is much worse than ResNet-10, 18 and 34 (likely due to overfitting, the same can be noted on the mini-ImageNet datase).

Table 4: Comparison of test accuracy among different learners for 1-shot and 5-shot classification on mini-ImageNet and tiered-ImageNet.

		mini-ImageNet		tiered-ImageNet	
Method	Predictor	1-shot	5-shot	1-shot	5-shot
SimpleShot	SimpleShot	61.22±0.21	81.00±0.14	66.86±0.21	85.50±0.14
SimpleShot [21]	ℓ_2+SVM	62.53±0.21	80.41±0.15	70.12±0.21	85.54±0.14
SimpleShot	ℓ_2+SVM+PA	64.77±0.21	79.68±0.15	71.26±0.21	87.34±0.14
S2M2	S2M2	64.93±0.18	83.18±0.11	66.33±0.05	81.44±0.09
S2M2 [11]	ℓ_2+SVM	66.54±0.20	83.51±0.13	69.43±0.22	85.27±0.15
S2M2	ℓ_2+SVM+PA	68.02±0.20	82.01±0.11	69.92±0.20	87.02±0.15

Table 5: 5-way 1-shot classification accuracy (%) on mini-ImageNet with different backbones.

	SimpleShot [21]	ℓ_2+SVM	ℓ_2+SVM+DA
ResNet-10	57.85±0.20	60.22±0.20	62.36±0.20
ResNet-18	60.16±0.20	61.47±0.20	63.21±0.20
ResNet-34	59.61±0.20	61.06±0.20	63.53±0.20
ResNet-50	53.20±0.20	55.76±0.20	56.73±0.20
MobileNet	59.33±0.20	60.17±0.20	61.49±0.20
WRN	61.22±0.21	62.53±0.21	64.77±0.21

Table 6: 5-way 1-shot classification accuracy (%) on tiered-ImageNet with different backbones.

	SimpleShot [21]	ℓ_2+SVM	ℓ_2+SVM+DA
ResNet-10	64.58±0.23	66.23±0.22	66.60±0.23
ResNet-18	68.64±0.22	69.68±0.22	69.97±0.23
ResNet-34	70.52±0.22	71.10±0.22	71.18±0.20
MobileNet	68.66±0.23	68.77±0.21	69.23±0.22
WRN	66.86±0.21	70.12±0.22	70.55±0.22

4.5 Comparison with Distribution Calibration [22]

Table 7: 5-way 1-shot and 5-way 5-shot classification accuracy (%) on mini-ImageNet, CUB, and tiered-ImageNet.

	mini-ImageNet		CUB		tiered-ImageNet	
Predictor	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
DC	68.57±0.55	82.88±0.42	79.56±0.87	90.67±0.35	70.42±0.25	87.24±0.41
ℓ_2+SVM	66.54±0.20	83.51±0.13	80.91±0.20	92.26±0.10	69.43±0.22	85.27±0.15
ℓ_2+SVM+PA	68.02±0.20	82.01±0.11	81.06±0.20	90.68±0.12	69.92±0.22	87.02±0.15

Distribution Calibration [22] applies the feature transformation, augmentation and a linear classifier simultaneously to achieve the performance improvement. However, their paper does not consider the individual impact of each component to validate if the gain comes from the Distribution Calibration step. Thus, we put such a method under scrutiny and conduct ablation experiments. We observe that ℓ_2+SVM outperforms DC in 3 out of 6 settings. In other settings, DC does not have a clear cut advantage either. We conclude that different factors of FSL should be disentangled before new papers stake claims about the gains they achieve.

5 Conclusions

In this paper, we investigate the predictor part of FSL that is often overlooked and not studied in enough detail by many authors. We show that the predictor part containing three components, that is, feature normalization, feature augmentation and classifier, can yield a very good performance by careful combination of these components. A simple FSL variant that achieves a competitive

performance with recent state-of-the-art methods on mini-ImageNet, tiered-ImageNet and CUB, without any fine-tuning.

References

[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918. IEEE, 2012.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[3] Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. Multi-level semantic feature augmentation for one-shot learning. *IEEE Transactions on Image Processing*, 28(9):4594–4605, 2019.

[4] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8059–8068, 2019.

[5] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[6] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[7] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *International Conference on Learning Representations*, 2019.

[8] Piotr Koniusz, Hongguang Zhang, and Fatih Porikli. A deeper look at power normalizations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5774–5783, 2018.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[10] Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004.

[11] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2218–2227, 2020.

[12] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[13] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

[14] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2017.

[15] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[17] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[18] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[19] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[20] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5495–5504, 2018.

[21] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[22] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. 2021.

[23] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020.

[24] Hongguang Zhang and Piotr Koniusz. Power normalizing second-order similarity network for few-shot learning. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 1185–1193. IEEE, 2019.

[25] Yinghui Zhang, Bo Sun, Yongkang Xiao, Rong Xiao, and YunGang Wei. Feature augmentation for imbalanced classification with conditional mixture wgans. *Signal Processing: Image Communication*, 75:89–99, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **[TODO]**
   (b) Did you describe the limitations of your work? **[TODO]**
   (c) Did you discuss any potential negative societal impacts of your work? **[TODO]**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[TODO]**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? **[TODO]**

(b) Did you include complete proofs of all theoretical results? **[TODO]**

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[TODO]**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? **[TODO]**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[TODO]**