Matching Feature Sets for Few-Shot Image Classification

Arman Afrasiyabi[⋆•], Hugo Larochelle[◇†•], Jean-François Lalonde[⋆], Christian Gagné[⋆†•]
[⋆]Université Laval, [◇]Google Brain, [†]Canada CIFAR AI Chair, [•]Mila

https://lvsn.github.io/SetFeat/

Abstract

In image classification, it is common practice to train deep networks to extract a single feature vector per input image. Few-shot classification methods also mostly follow this trend. In this work, we depart from this established direction and instead propose to extract sets *of feature vectors for each image. We argue that a set-based representation intrinsically builds a richer representation of images from the base classes, which can subsequently better transfer to the few-shot classes. To do so, we propose to adapt existing feature extractors to instead produce* sets *of feature vectors from images. Our approach, dubbed SetFeat, embeds shallow self-attention mechanisms inside existing encoder architectures. The attention modules are lightweight, and as such our method results in encoders that have approximately the same number of parameters as their original versions. During training and inference, a set-to-set matching metric is used to perform image classification. The effectiveness of our proposed architecture and metrics is demonstrated via thorough experiments on standard few-shot datasets—namely miniImageNet, tieredImageNet, and CUB—in both the 1- and 5-shot scenarios. In all cases but one, our method outperforms the state-of-the-art.*

1. Introduction

The task of few-shot image classification is to transfer knowledge gained on a set of "base" categories, assumed to be available in large quantities, to another set of "novel" classes of which we are given only very few examples. To solve this problem, a popular strategy is to employ a deep feature extractor which learns to convert an input image into a feature vector that is both discriminative and transferable to the novel classes. In this context, the common practice is to train a model to extract, for a given input, a *single* feature vector from which classification decisions are made.

In this paper, we depart from this established strategy by proposing instead to represent images as *sets* of feature vectors. With this, we aim at learning a richer feature space that is both more discriminative and easier to transfer to the novel domain, by allowing the network to focus on different characteristics of the image and at different scales. The intuition motivating that approach is that decomposing the representation into independent components should allow the capture of several distinctive aspects of images that can then be used efficiently to represent images of novel classes.

To do so, we take inspiration from Feature Pyramid Networks [34] which proposes to concatenate multi-scale feature maps from convolutional backbones. In contrast, however, we do not just poll the features themselves but rather embed shallow self-attention modules (called "mappers") at various scales in the network. This adapted network therefore learns to represent an image via a set of attention-based latent representations. The network is first pre-trained by injecting the signal of a classification loss at every mapper. Then, it is fine-tuned in a meta-training stage, which performs classification by computing the distance between a query (test) and a set of support (training) samples in a manner similar to Prototypical Networks [49]. Here, the main difference is that the distance between samples is computed using set-to-set metrics rather than traditional distance functions. To this end, we propose and experiment with three set-to-set metrics.

This paper presents the following contributions. First, it presents the idea of reasoning on *sets* and a set-based inference of feature vectors extracted from images. It shows that set representation yields improved performance on few-shot image classification without increasing the total number of network parameters. Second, it presents a straightforward way to adapt *existing* backbones to make them extract *sets* of features rather than single ones, and processing them in order to achieve decisions. To do so, it proposed to embed simple self-attention modules in between convolutional blocks, with examples of adapted three popular backbones, namely Conv4-64, Conv4-256, and ResNet12. It also proposes set-to-set metrics for evaluation of differences between query and support set. Third, it presents extensive experiments on three few-shot datasets, namely miniImageNet, tieredImageNet and CUB. In almost all cases, our method outperforms the state-of-the-art. Notably, our method gains 1.83%, 1.42%, and 1.83% accuracy in 1-shot over the baselines in miniImageNet, tieredImageNet and CUB, respectively.

2. Related work

Our work falls within the domain of inductive few-shot image classification [17,43,49,58], unlike transductive methods [6, 12, 28, 72] which exploit the structural information of the entire novel set. The following covers the most relevant works under few-shot learning research area, and beyond.

Training framework Two main training frameworks have been explored so far, namely meta learning or standard transfer learning. On one hand, meta learning [17, 43, 49, 58], also named episodic training, repeatedly samples small subsets of base classes to train the network, thereby simulating few-shot "episodes" during training. For example, some methods (e.g. [4, 17, 68]) aim at training a model to classify the novel classes with a small number of gradient updates. On the other hand, standard transfer learning methods [1, 8, 22, 41, 54] usually rely on a generic batch training with a metric-based (such as margin-based) criteria. Recently, several works [65, 67, 73] have shown that combining both standard transfer learning, following by a second meta-training stage can offer good performance. We employ a similar two-stage training procedure in this paper.

Metrics Metric-based approaches [5, 20, 29, 32, 33, 40, 45, 49, 52, 55, 58, 62, 73, 76] aim at improving how the distance is calculated for better performance at training and inference. In this aspect, our work is related to ProtoNet [49] as it also seeks to reduce the distance between a query and the centroid of a set of support examples of the corresponding class, while differing by proposing the use of *set-to-set* distance metrics for computing distance over several feature vectors. Other highly related works include FEAT [67], CTX [13], TapNet [69] and ConstellationNet [66], which apply attention embedding adaptation functions on the episodes before computing the distance between query and the prototypes of the support set. Unlike them, our method extracts a set of feature vectors given a query and support set, over which a set-to-set metric is applied for computing the distances.

Extra data Relying on extra data [10, 11, 19, 21, 23, 25, 36, 37, 44, 46, 60, 61, 74, 75, 77] is another strategy for building a well-generalized model. The augmented data can be in the form of hallucination with a data generator function [25, 60], using unlabelled data under semi-supervised [44, 70] or self-supervised [21, 50] frameworks, or aligning the novel classes to the base data [1]. In contrast, our approach does not require any additional data beyond the base classes.

Vision transformers Our method employs shallow attention mappers that are inspired by the multi-head attention mechanism proposed in [57] and adapted to images by Dosovitskiy *et al*. [14]. In contrast to these works, our feature mappers are independent, are shallow (thus lightweight), are not unified by FC-layers, and can extend to convolutions as in [24, 63]. We also employ several independent mappers at different depths/scales in the network.

Feature sets Feature sets have long been investigated in computer vision [27, 30, 42]. In the more recent deep learning literature, our method bears resemblance to FPN [34] which extracts multi-scale features for object detection, and deep sets [71] which proposed permutation-invariant networks that operate on *input* sets. In contrast, our work computes feature sets to tackle few-shot image classification. From the few-shot perspective, our work is related to the transductive approach of [28], which employs the unlabeled query set to augment the support set, and [73] which uses Earth Mover's Distance over the representations of multi-cropped images with a generic data augmentation. Here, we focus on the extraction of feature sets for each support example in an inductive setting.

3. Preliminaries

In N-way K-shot (where K is small) image classification, we aim to predict the class of a given query example $\mathbf{x}_q$ using a support set $\mathcal{S}$ containing K examples for each of the N different classes considered. Let $\mathcal{S}^n \in \mathcal{S}$ and $\mathcal{S}^n = \{(\mathbf{x}_i^n, y_i = n)\}_{i=1}^K$ be a set of example-label pairs, all pairs of that set $\mathcal{S}^n$ belonging to class n. In addition, let $f(\mathbf{x}|\theta^f)$ be a convolutional feature extractor composed of B blocks parameterized by $\theta^f = \{\theta_b^f\}_{b=1}^B$, where θ_b^f are the parameters of the b-th block. Here, a "block" broadly refers to a group of convolutional layers (with or without skip links), typically followed by a downscaling operation reducing the features spatial dimensions (*e.g.*, pooling). The features after a given block b can be obtained from $\mathbf{z}_b \equiv f(\mathbf{x}|\{\theta_i^f\}_{i=1}^b)$.

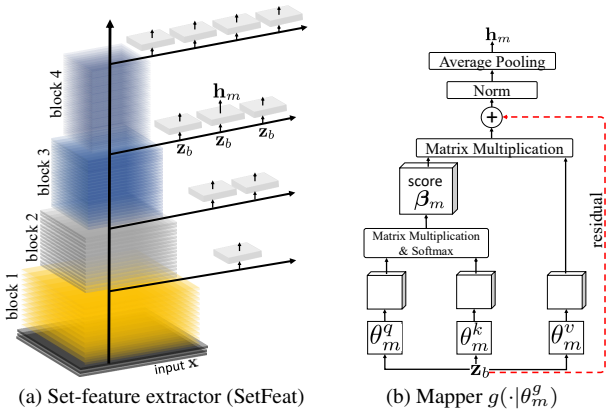
In this work, we introduce a *set-feature* extractor, dubbed "SetFeat", which extracts a set of M feature vectors from images, rather than a single vector as it is typically done in the literature [17, 49, 58]. Formally, SetFeat produces the set $\mathcal{H} = \{\mathbf{h}_m\}_{m=1}^M$ of M feature vectors $\mathbf{h}_m$ through shallow self-attention mappers, and employs set-to-set matching metrics to establish the similarity between images in set-feature space. The following section presents our approach in detail.

4. Set-based few-shot image classification

In this section, we first discuss our proposed set-feature extractor SetFeat, then dive into the details of our proposed set-to-set metrics. Finally, our proposed inference and training procedures are presented.

4.1. Set-feature extractor

The overall architecture of SetFeat is illustrated in fig. 1. As mentioned in sec. 3, its goal is to map an input image



(a) Set-feature extractor (SetFeat) (b) Mapper $g(\cdot|\theta^g_m)$

Figure 1. The schematic overview of the proposed set-feature extractor (SetFeat) and detail of a single attention-based mapper: (a) given an input $\mathbf{x}$, SetFeat first extracts (convolutional) feature vectors $\mathbf{z}_b$ at each of its blocks, while at each block attention-based mappers (illustrated as small rectangles) convert $\mathbf{z}_b$ into a different embedding $\mathbf{h}_m$; (b) a single mapper m at block b extracts embedding $\mathbf{h}_m$ using an attention mechanism containing query θ^q_m and key θ^k_m to build attention scores $\boldsymbol{\beta}_m$, with self-attention inferred using value θ^v_m and score $\boldsymbol{\beta}_m$. This work focuses on backbones made of $B = 4$ blocks, consistent with popular few-shot image classification backbones such as Conv4 [58] and ResNet [26].

$\mathbf{x}$ to a feature set $\mathcal{H}$. To this end, and inspired by [14, 57], we embed segregated self-attention mappers $g(\cdot)$ throughout the network, as shown in fig. 1a. We reiterate (*cf*. sec. 2), however, that our mappers are different from multi-head attention-based models [14, 57] for two main reasons. First, each mapper in our approach is composed of a single attention head, thus we do not rely on fully connected layers to concatenate multi-head outputs. Our feature mappers are therefore separate from each other and each extract their own set of features. Second, our feature mappers are shallow (unit depth), with the learning mechanisms relying on the convolutional layers of the backbone.

The detail of the m-th feature mapper $g(\mathbf{z}_{b_m}|\theta^g_m)$, where b_m represents the block preceding the mapper, is illustrated in fig. 1b. The learned representation $\mathbf{z}_{b_m} \in \mathbb{R}^{P\times D^p}$ is separated into P non-overlapping patches of D^p dimensions. In this work, we use patches of size 1×1, each patch is therefore a 1-D vector of D^p elements. Following Vaswani *et al*. [57], an attention map is first computed using two parameterized elements $q(\mathbf{z}_{b_m}|\theta^q_m)$ and $k(\mathbf{z}_{b_m}|\theta^k_m)$:

$$\boldsymbol{\beta}_m = \text{Softmax}\left(q(\mathbf{z}_{b_m}|\theta^q_m)k(\mathbf{z}_{b_m}|\theta^k_m)^\top/\sqrt{d_k}\right), \quad (1)$$

where $\boldsymbol{\beta}_m \in \mathbb{R}^{P\times P}$ is the attention score over the patches of $\mathbf{z}_{b_m}$, and $\sqrt{d_k}$ is the scaling factor. Then, we compute the dot-product attention over the patches of $\boldsymbol{\beta}_m$ using $v(\mathbf{z}_{b_m}|\theta^v_m)$ in the following form:

$$\mathbf{a}_m = \boldsymbol{\beta}_m\, v(\mathbf{z}_{b_m}|\theta^v_m)\,, \quad (2)$$

where $\mathbf{a}_m \in \mathbb{R}^{P\times D^a}$ consists of P patches of D^a dimensions and D^a is the dimension of z_b. If the backbone feature extractor is ResNet [26] (see sec. 5.1), we add a residual to the computed attention ($\mathbf{a}_m + \mathbf{z}_{b_m}$). In this case, if the dimensions mismatch ($D^a \neq D^p$), we use 1×1 convolution of unit stride and kernel size similar to downsampling. Finally, the feature vector $\mathbf{h}_m$ is computed by taking the mean of over the patches (over the P dimension).

4.2. Set-to-set matching metrics

Having covered how SetFeat extracts a feature set for each input instance to process, we now proceed to how it leverages this set for image classification. In this context, we need to compare the feature set of the query with the feature sets corresponding to each instance of the support set of each class, to infer the class of the query. More specifically, in order to proceed with a distance-based approach as we do with prototypical networks, we need a set-to-set metric allowing the measure of distance over sets. We now present three distinct set-to-set metrics $d_{\text{set}}(\mathbf{x}_q, \mathcal{S}^n)$, which measure the distance between multiple feature sets, where $\mathbf{x}_q$ is the query, and $\mathcal{S}^n$ is a support set for class n (*cf*. sec. 3). We employ the shorthand $\mathbf{h}_m(\mathbf{x}) \equiv g_m(\mathbf{z}_{b_m}|\theta^g_m)$ to refer to a feature extracted by mapper m. In addition, we also define $\bar{\mathbf{h}}_m(\mathcal{S}) \equiv \frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}\in\mathcal{S}}\mathbf{h}_m(\mathbf{x})$ as the centroid of features extracted by mapper m on the support set $\mathcal{S}$. The following set metrics are built upon a generic distance function $d(\cdot,\cdot)$. In practice, we employ the negative cosine similarity function, *i.e*., $d(\cdot,\cdot) = -\cos(\cdot,\cdot)$.

Match-sum aggregates the distance between matching mappers for the query and supports:

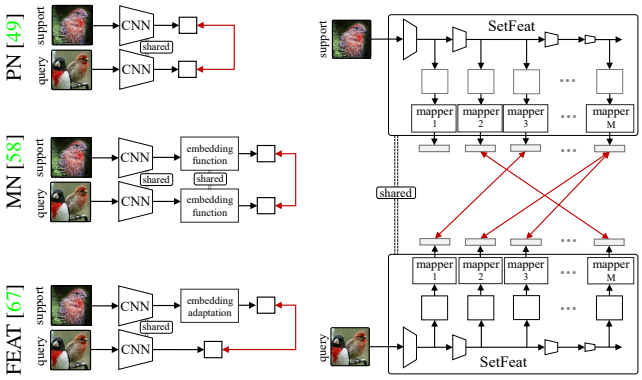
$$d_{\text{ms}}(\mathbf{x}_q, \mathcal{S}^n) = \sum_{i=1}^{M} d\left(\mathbf{h}_i(\mathbf{x}_q), \bar{\mathbf{h}}_i(\mathcal{S}^n)\right). \quad (3)$$

We use this metric as a baseline, as it parallels a common strategy of building representations simply by concatenating several feature vectors and invoking a standard metric on the flattened feature space.

Min-min uses the minimum distance across all possible pairs of elements from the query and support set centroids:

$$d_{\text{mm}}(\mathbf{x}_q, \mathcal{S}^n) = \min_{i=1}^{M}\min_{j=1}^{M} d\left(\mathbf{h}_i(\mathbf{x}_q), \bar{\mathbf{h}}_j(\mathcal{S}^n)\right). \quad (4)$$

Such a metric leverages directly the set structure of features.



(a) Three baseline methods (b) Set matching with sum-min

Figure 2. Illustration of 1-shot image classification using (a) three existing methods and (b) our approach with the sum-min metric. (a) Given a query and support, existing methods either directly match the query to support (ProtoNet (PN) [49]), apply a single embedding function over both support and query (MatchingNetwork (MN) [58]), or perform embedding adaptation on the support before matching it with the query (FEAT [67]). (b) Our SetFeat method extracts sets of features for both of the support and query, which are then processed by the self-attention mappers. The set metric is then computed over the embeddings.

Sum-min departs from the min-min metric by aggregating with a sum the minimum distances between the mappers computed on query and support set centroids:

$$d_{\text{sm}}(\mathbf{x}_q, \mathcal{S}^n) = \sum_{i=1}^{M} \min_{j=1}^{M} d\left(\mathbf{h}_i(\mathbf{x}_q), \bar{\mathbf{h}}_j(\mathcal{S}^n)\right) . \quad (5)$$

A schematic illustration of the sum-min metric is shown in fig. 2, which also illustrates its difference with respect to three baseline few-shot models. Our method is different from FEAT [67] (and MN [58]) in two main ways. First, we define sets over features extracted from each example while FEAT/MN do so over the support set directly. In an extreme 1-shot case, the FEAT "set" degenerates to a single element (1 support). Beneficial for few-shot, our work always keeps sets of many elements, regardless of the support set cardinality. Second, our method employs the parameterized mappers for set feature extraction. Here, we adjust the backbone (unlike FEAT and MN) so that adding mappers results in the same total number of parameters. Third, our method employs non-parametric set-to-set metrics, used for inference.

4.3. Inference

Given one of our metrics $d_{\text{set}} \in \{d_{\text{ms}}, d_{\text{mm}}, d_{\text{sm}}\}$ defined in the previous section, we follow the approach of Prototypical Networks [49] with SetFeat and model the probability of a query example $\mathbf{x}_q$ belonging to class $y = n$, where

Algorithm 1: SetFeat meta-training and validation.

Data: Network parameterized by $\theta = \{\theta^f, \theta^g\}$ made of a backbone of B convolution blocks ($\theta^f = \{\theta^f_b\}_{b=1}^B$) and M mappers ($\theta^g = \{\theta^g_m\}_{m=1}^M$); episodic train dataset $\mathcal{X}_{\text{train}}$ containing episodes of support set $\mathcal{S}$ and a query example $\mathbf{x}_q$; validation dataset $\mathcal{X}_{\text{valid}}$; maximum number of epochs $t^{\max}$; 0-1 loss function ℓ_{0-1} used to measure the validation accuracy.

Result: Best model defined as $\theta_{\text{best}} = \{\theta^f_{\text{best}}, \theta^g_{\text{best}}\}$

$E^{\text{best}}_{\text{valid}} \leftarrow \infty$

for $t = 1, \ldots, t^{\max}$ **do**

  for $(\mathbf{x}_q, \mathcal{S}) \in \mathcal{X}_{\text{train}}$ **do**

    $\ell^t \leftarrow -\log p(y_q|\mathbf{x}_q, \mathcal{S})$ using eq. 6

    Update network θ with backpropagation of loss ℓ^t

  end

  $\hat{y}_q \leftarrow \arg\min_{\mathcal{S}^n \in \mathcal{S}} d_{\text{set}}(\mathbf{x}_q, \mathcal{S}^n), \ \ \forall (\mathbf{x}_q, \mathcal{S}) \in \mathcal{X}_{\text{valid}}$

  $E_{\text{valid}} \leftarrow \frac{1}{|\mathcal{X}_{\text{valid}}|} \sum_{(\mathbf{x}_q, \mathcal{S}) \in \mathcal{X}_{\text{valid}}} \ell_{0-1}(\hat{y}_q, y_q)$

  if $E_{\text{valid}} < E^{\text{best}}_{\text{valid}}$ **then**

    $E^{\text{best}}_{\text{valid}} \leftarrow E_{\text{valid}}$

    $\theta_{\text{best}} \leftarrow \theta$

  end

end

$n \in \{1, \ldots, C\}$ (N-way), using a softmax function:

$$p(y = n|\mathbf{x}_q, \mathcal{S}) = \frac{\exp(-d_{\text{set}}(\mathbf{x}_q, \mathcal{S}^n))}{\sum_{\mathcal{S}^i \in \mathcal{S}} \exp(-d_{\text{set}}(\mathbf{x}_q, \mathcal{S}^i))} , \quad (6)$$

with $\mathcal{S}$ as the (few-shot) support set.

4.4. Training procedure

We follow recent literature [65, 67, 73] and leverage a two-stage procedure to train SetFeat using one of our proposed set-to-set metrics. The first stage performs standard pre-training where a random batch $\mathcal{X}_{\text{batch}}$ of instances $\mathbf{x}$ from base classes are drawn from the training set. Here, we append fully-connected (FC) layers $\mathbf{o}_m$ to convert each of the mapper features $\mathbf{h}_m$ into logits in order to achieve classification over the C classes. From that, cross-entropy loss is used to train each mapper independently:

$$\ell_{\text{pre}} = -\sum_{\mathbf{x}_i \in \mathcal{X}_{\text{batch}}} \sum_{m=1}^{M} \log \frac{\exp(o_{m,y_i}(\mathbf{h}_{m,i}))}{\sum_{c=1}^{C} \exp(o_{m,c}(\mathbf{h}_{m,i}))} , \quad (7)$$

where $o_{m,c}$ is the FC layer output of mapper m for class c, $\mathbf{h}_{m,i}$ is the feature set of mapper m for instance $\mathbf{x}_i$, and y_i is the target output corresponding to instance $\mathbf{x}_i$.

The second stage discards the FC layers that were added in the first stage, and employs episodic training [49, 58] which simulates the few-shot scenario on the base training dataset. This stage is presented in algorithm 1. Specifically, we randomly sample N-way K-shot and Q-queries, then we compute the probability scores for each query using eq. 6. Finally, we update the parameters of the network after computing the cross-entropy loss.

5. Evaluation

This section first covers the details of our experiments with SetFeat, which are based on conventional backbones employed in the few-shot image classification literature. This is followed by description of the datasets and implementation details are described next. Finally, we present the evaluations of SetFeat with our set-matching metrics using four backbones with three datasets.

5.1. Backbones

We adopt the following three popular backbones, each composed of four blocks: (a) Conv4-64 [58], which consists of 4 convolution layers with 64/64/64/64 filters for a total of 0.113M parameters, (b) Conv4-512 [58]: 96/128/256/512 for 1.591M parameters, and (c) ResNet12 [26, 40]: 64/160/320/640 for 12.424M parameters. In all experiments below, we embed a total of 10 self-attention mappers throughout each backbone by following this per-block pattern: 1 mapper after block 1, then 2, 3 and 4 mappers for the three subsequent blocks. We experiment with other choices of mapper configurations in sec. 6.3.

Since our attention-based feature mappers require additional parameters, we correspondingly reduce the number of kernels in the backbone feature extractors to ensure that the performance gains are not simply due to the over-parameterization. Specifically, our SetFeat4-512, the counterpart of Conv4-512, uses a reduced set of 96/128/160/200 convolution kernels for a total of 1.583M parameters (compared to 1.591M for Conv4-512). SetFeat12, counterpart of ResNet12, consists of 128/150/180/512 kernels for 12.349M parameters (comp. 12.424M for ResNet12). For Conv4-64, reducing the amount of parameters collapses the training (as noted in [16, 64, 67]) since it contains very few parameters already. Our SetFeat4-64 therefore has more parameters (0.238M vs 0.113M for Conv4-64), but in sec. 6.2 we artificially augment the number of parameters for Conv4-64 and show our approach still outperforms it.

Convolutional attention [63] is used in SetFeat4-512 and SetFeat12. Particularly, we used single depth convolution and batch normalization to parameterize key, query and value in each mapper. The output dimension of the feature mappers is set to the number of channels in the last layer of the feature extractor — having all mappers producing feature vectors of the same dimension is a necessary condition for our proposed metrics. For SetFeat4, FC-layers are used to compute the attention in order to limit the number of additional parameters as much as possible. The supplementary material includes the details of our implementation.

5.2. Datasets and implementation details

We conduct experiments on miniImageNet [58] (100/50/50 train/validation/test classes), tieredImageNet [44] (351/97/160) for object recognition, and CUB [59] (100/50/50) for fine-grained classification. To pretrain SetFeat4, we used Adam [40] with a learning rate (lr) of 0.001 and weight decay of 5×10^{-4}. Batch size is fixed to 64. For

Table 1. Evaluation on miniImageNet in 5-way. Bold/blue is best/second, and $\pm$ is the 95% confidence intervals in 600 episodes.

	Method	Backbone	1-shot	5-shot
	ProtoNet [49]	Conv4-64	49.42±0.78	68.20±0.66
	MAML [18]	Conv4-64	48.07±1.75	63.15±0.91
	RelationNet [52]	Conv4-64	50.44±0.82	65.32±0.70
	Baseline++ [8]	Conv4-64	48.24±0.75	66.43±0.63
	IMP [3]	Conv4-64	49.60±0.80	68.10±0.80
	MemoryNet [7]	Conv4-64	53.37±0.48	66.97±0.35
	Neg-Margin [35]	Conv4-64	52.84±0.76	70.41±0.66
	MixtFSL [2]	Conv4-64	52.82±0.63	70.67±0.57
	FEAT [67]	Conv4-64	55.15±0.20	71.61±0.16
	MELR [16]	Conv4-64	55.35±0.43	72.27±0.35
	BOIL [39]	Conv4-64	49.61±0.16	66.45±0.37
– Ours –	Match-sum	SF4-64	55.74±0.65	72.18±0.70
– Ours –	Min-min	SF4-64	56.22±0.89	72.70±0.65
– Ours –	Sum-min	SF4-64	**57.18**±0.89	**73.67**±0.71
	ProtoNet† [49]	Conv4-512	53.52±0.43	73.34±0.36
	MAML [17]	Conv4-512	49.33±0.60	65.17±0.49
	Relation Net [52]	Conv4-512	50.86±0.57	67.32±0.44
	PN+rot [22]	Conv4-512	56.02±0.46	74.00±0.35
	CC+rot [21]	Conv4-512	56.27±0.43	74.30±0.33
	MELR [16]	Conv4-512	57.54±0.44	74.37±0.34
– Ours –	Match-sum	SF4-512	56.50±0.85	72.69±0.68
– Ours –	Min-min	SF4-512	58.57±0.87	73.46±0.68
– Ours –	Sum-min	SF4-512	**59.10**±0.87	**74.97**±0.66
	AdaResNet [38]	ResNet12	56.88±0.62	71.94±0.57
	TADAM [40]	ResNet12	58.50±0.30	76.70±0.30
	MetaOptNet [31]	ResNet12	62.64±0.61	78.63±0.46
	Neg-Margin [35]	ResNet12	63.85±0.76	81.57±0.56
	MixtFSL [2]	ResNet12	63.98±0.79	82.04±0.49
	Meta-Baseline [9]	ResNet12	63.17±0.23	79.26±0.17
	Distill [54]	ResNet12	64.82±0.60	82.14±0.43
	DeepEMD [73]	ResNet12	65.91±0.82	82.41±0.56
	DMF [65]	ResNet12	67.76±0.46	82.71±0.31
	MELR [16]	ResNet12	67.40±0.43	**83.40**±0.28
	ProtoNet§ [49]	ResNet12	62.39	80.53
	FEAT§ [67]	ResNet12	66.78	82.05
– Ours –	Match-sum	SF-12 -	67.41±0.64	81.79±0.55
– Ours –	Min-min	SF-12 -	67.88±0.55	82.07±0.61
– Ours –	Sum-min	SF-12 -	**68.32**±0.62	82.71±0.46

§confidence interval not provided † taken from [16]

Mappers dimension: SF4-64$\in \mathbb{R}^{64}$, SF4-512$\in \mathbb{R}^{200}$, SF12$\in \mathbb{R}^{512}$

Table 2. TieredImageNet evaluation. Bold/red is best/second best, and ± indicates the 95% conf. intervals over 600 episodes of 5-way.

	Method	Backbone	1-shot	5-shot
	OptNet [31]	ResNet12	65.99±0.72	81.56±0.53
	MTL [51]		65.62±1.80	80.61±0.90
	DNS [47]		66.22±0.75	82.79±0.48
	Simple [54]		69.74±0.72	84.41±0.55
	TapNet [69]		63.08±0.15	80.26±0.12
	ProtoNet† [49]		68.23± 0.23	84.03±0.16
	FEAT [67]		70.80±0.23	84.79±0.16
	MixtFSL [2]		70.97±1.03	86.16±0.67
	Distill [54]		71.52±0.69	86.03±0.49
	DeepEMD [73]		71.16±0.87	86.03±0.58
	DMF [65]		71.89±0.52	85.96±0.35
	MELR [16]		72.14±0.51	87.01±0.35
	Distill [45]		72.21±0.90	87.08±0.58
– Ours –	Match-sum	- SF12 -	71.22±0.86	85.43±0.55
	Min-min		71.75±0.90	86.40±0.56
	Sum-min		**73.63**±0.88	**87.59**±0.57

†taken from [31]; Mappers dimension: SF12 $\in \mathbb{R}^{512}$

Table 3. Fine-grained evaluation using CUB in 5-way. ± is the 95% confidence intervals on 600 episodes(‡taken from [53]).

	Method	Backbone	1-shot	5-shot
	MatchingNet [58]	Conv4-64	61.16±0.89	72.86±0.70
	ProtoNet [49]		64.42±0.48	81.82±0.35
	MAML [17]		55.92±0.95	72.09±0.76
	RelationNet [52]		62.45±0.98	76.11±0.69
	FEAT [67]		68.87±0.22	82.90±0.15
	MELR [16]		70.26±0.50	85.01±0.32
– Ours –	Match-sum	SF4-64	67.35±0.93	83.82±0.61
	Min-min		70.15±0.93	84.94±0.64
	Sum-min		**72.09**±0.92	**87.05**±0.58
	Robust-20 [15]	ResNet18	58.67±0.7	75.62±0.5
	RelationNet‡ [52]		67.59±1.0	82.75±0.6
	MAML‡ [17]		68.42±1.0	83.47±0.6
	ProtoNet‡ [49]		71.88±0.9	86.64±0.5
	Baseline++ [8]		67.02±0.9	83.58±0.5
	MixtFSL [2]		73.94±1.1	86.01±0.5
	Neg-Margin [35]		72.66±0.9	89.40±0.4
– Ours –	Match-sum	- SF12*-	77.95± 0.83	88.93± 0.49
	Min-min		78.51±0.82	89.73±0.47
	Sum-min		**79.60**±0.80	**90.48**± 0.44

Mappers dimension: SF4-64 $\in \mathbb{R}^{64}$, and SF12* $\in \mathbb{R}^{480}$

SetFeat12, we used Nesterov momentum with an initial lr of 0.1, momentum of 0.9 and weight decay of 5×10^{-4}. We follow [65, 67, 73] for normalization and data augmentation. In the meta-training stage, SGD is used for all architectures. Validation sets are used to tune the schedule of the optimizer.

5.3. Quantitative and comparative evaluations

miniImageNet Table 1 presents evaluations of SetFeat with our set-to-set metrics on the miniImageNet dataset. First, we observe that our sum-min metric outperforms both the other proposed metrics and the state-of-the-art except in the 5-shot with SetFeat12. In particular, SetFeat4-64 (sum-min) results in an accuracy gain of 1.83% and 1.4% over MELR [16] in 1- and 5-shot, respectively.

tieredImageNet Table 2 presents the tieredImageNet evaluation of SetFeat12 with our proposed metrics. Our sum-min metric results in 1.42% and 0.51 % improvement over the baseline Distill [45] in 1- and 5-shot. Please note that baselines such as Distill [45], MELR [16], and FEAT [67] contain more parameters than the original ResNet12 and SetFeat12.

CUB Table 3 illustrates the fine-grained classification evaluation of our approach, compared to Conv4-64 and ResNet18. We observe that SetFeat4-64 (min-min) again surpasses all baselines by providing gains of 1.83% and 2.04% over MELR [16] in 1- and 5-shot respectively. When comparing with ResNet18, we further reduce the number of convolution kernels to 128/150/196/480 (dubbed SetFeat12*) to better match the number of parameters (11.466M for SetFeat12* vs 11.511M for ResNet18). Our approach again defines a new state-of-the-art performance in this scenario.

6. Ablation

In this section, we further analyze SetFeat to explore alternative design decisions and gain a better understanding as to why our set-based model achieves better accuracy.

6.1. Mapper configurations

We now experiment with different ways of embedding ten mappers throughout the backbone levels. We compare: 1) putting all mappers on the last layer (0-0-0-10); 2) a single mapper per block (1-1-1-1); 3) distributing mappers more equally (2-2-3-3); and 4) employing a progressive growth strategy (1-2-3-4) (this last one being used in the main evaluation in sec. 5). Table 4 compares these four strategies on both SetFeat4-64 and SetFeat4-512 on the validation set of miniImageNet. We observe that placing mappers throughout the network yields better results than putting them all at the end. The two other options perform similarly. We also observe that (2-2-3-3) only beats (1-2-3-4) using shallower network SetFeat4-64 in 5-shot. Otherwise, progressive growth either reaches or surpasses the other combinations. Note, that going from 0-0-0-10 to 1-2-3-4 or 2-2-3-3 improves performance

Table 4. Ablation of different mapper-level combinations using miniImageNet. The results are validation accuracy with min-sum.

Mappers	SetFeat4-64 1-shot	SetFeat4-64 5-shot	SetFeat4-512 1-shot	SetFeat4-512 5-shot
ProtoNet*	53.51	71.57	–	–
0-0-0-1	53.55	71.51	–	–
1-2-3-4 (concat)	53.56	71.82	–	–
1-1-1-1	51.11	69.41	53.57	71.60
0-0-0-10	52.90	69.49	55.36	71.59
2-2-3-3	54.73	71.98	56.29	74.74
1-2-3-4	54.71	71.35	58.74	75.30

* with Conv4-512

Table 5. Ablation of our SetFeat with miniImageNet and CUB on 600 episodes with augmented Conv4-64 and SetFeat4-64 in 5-way.

Method	miniImageNet 1-shot	miniImageNet 5-shot	CUB 1-shot	CUB 5-shot
ProtoNet [49]	49.42	68.20	68.23	84.03
ProtoNet* [49]	49.98	69.53	69.11	85.27
Sum-min (ours)	57.18	73.67	73.50	87.61

* our implementation with augmented Conv4-64

while using the same number of mappers, which confirms that multi-scale indeed helps. Additionally, removing our set-based representation by concatenating all mappers outputs and treating the result as a single (multi-scale) feature vector ("concat" in tab. 4) completely cancels any performance gain. Therefore, we conclude that it is our sets of multi-scale features that explains the performance improvement.

6.2. Over-parameterization of SetFeat4-64

Sec. 5.1 mentioned that the number of kernels in backbone feature extractors was reduced in such a way that adding our proposed attention-based mappers did not significantly change the total number of parameters in the network—but unfortunately doing so for Conv4-64 resulted in poor generalization as each of its four blocks is only composed of a single layer with 64 kernels. Here, we instead *augment* Conv4-64 and add parameters with three FC layers (of 512, 160, 64 dimensions) after the convolutional blocks. This reaches 0.239M parameters, which matches the 0.238M parameters of SetFeat4-64. Results are presented in table 5. Although the augmented Conv4-64 improves over the baseline Conv4-64, the improvements are significantly below those obtained by SetFeat4-64, showing that the additional parameters alone do not explain the performance gap.

6.3. Probing the activation of mappers

Let us now investigate whether all mappers are actually useful by analyzing the behavior under the sum-min metric

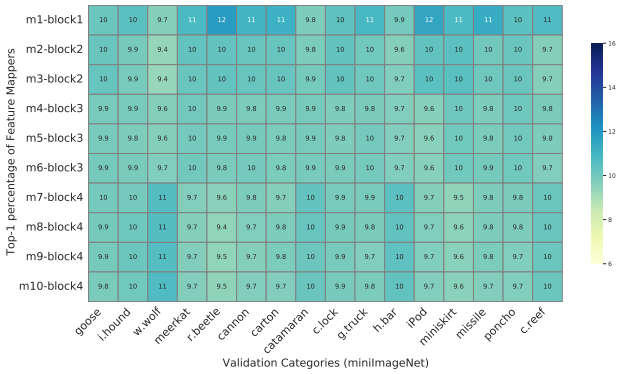
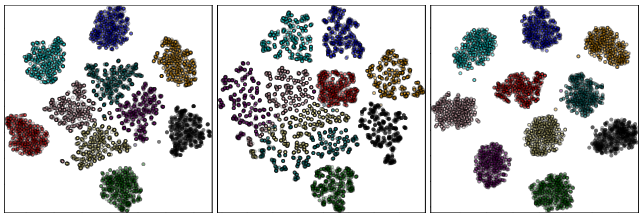


Figure 3. The percentage time each of the mappers (y-axis) is selected for each of the 16 validation categories (x-axis) of the miniImageNet dataset. The result is obtained by SetFeat12 and averaged over 600 episodes of 5-way 1-shot. While the earlier mappers are more often active, all mappers are consistently useful.



(a) miniImageNet (b) CUB (c) tieredImageNet

Figure 4. Visualizing mappers with t-SNE [56] on 640 randomly-sampled from validation set for (a) miniImageNet with SetFeat12, (b) CUB with SetFeat12* (sec. 5.3) and (c) tieredImageNet with SetFeat12. Points are color-coded according to the mapper.

(sec. 4.2). For this, fig. fig. 3 illustrates the percentage of time where a specific mapper (y-axis) provides the minimum prototype-query distance for each validation class (x-axis) in the miniImageNet dataset. This illustrates that low-level mappers are often active like the high-level ones, but all mappers are consistently being used across all validation classes, thereby validating that our proposed set-based representation is effective and working as expected.

In addition, fig. 4 shows t-SNE [56] visualizations of 640 embedded examples from miniImageNet, CUB, and tieredImageNet datasets using our set-feature extractor. Note how the distributions of mapper embeddings are generally disjoint and do not collapse to overlapping points, which shows intuitively that mappers extract different features.

6.4. Top-m analysis

The min-min and sum-min metrics (eqs. (4) and (5) respectively) are two ends of the spectrum: min-min takes the minimum distance across all mappers, while sum-min computes the sum over all the mappers. Here, we sort the mappers according to distance and sum the top-m as an

Table 6. Ablation of top-m mapper in the min-sum metric using SetFeat4 and SetFeat12* on CUB. The results are validation set.

Method	SetFeat4		SetFeat12*	
	1-shot	5-shot	1-shot	5-shot
top-1 (min-min)	70.15	84.94	78.51	89.73
top-2	70.84	85.30	77.92	89.87
top-4	70.34	85.95	78.37	89.78
top-8	71.47	86.88	79.56	90.03
top-10 (sum-min)	**72.09**	**87.05**	**79.60**	**90.48**

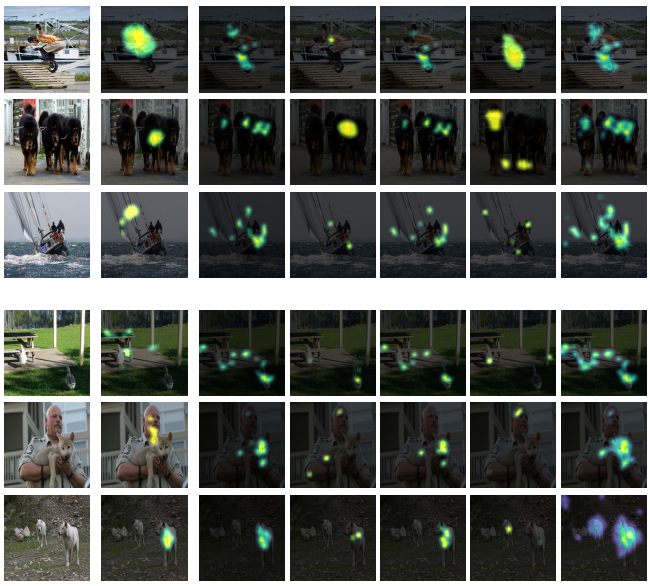
ablation shown in table 6. In general, we observe that the classification results progressively improve as we move towards sum-min, which uses all of the mappers.

6.5. Visualizing mappers saliency

We now visualize in fig. fig. 5 the impact of learning a set of features by visualizing the saliency map of each mapper, and by comparing them with the saliency maps of the single-feature approach of Chen *et al*. [8]. We compute the smoothed saliency maps [48] by single back-propagation through a classification layer. It can be seen that our approach devotes attention to many more parts of the images than when a single feature vector is learned. For example, note how a single dog is highlighted (fourth row of fig. fig. 5), whereas our mappers jointly fire on all three. Please consult the supplementary materials for more examples.

7. Discussion

This paper proposes to extract and match *sets* of feature vectors for few-shot image classification. This contrasts with the use of a monolithic single-vector representation, which is a popular strategy in that context. To produce these sets, we embed shallow attention-based mappers at different stages of conventional convolutional backbones. These mappers aim at extracting distinct sets of features with random initialization, capturing different properties of the images seen. Here, the non-linearity in the sum-min and min-min creates diversity: the inner minimum distance causes a non-linearity that forces the selection of a given mapper. Match-sum, our worst metric, only benefits from random initialization. We then rely on set-to-set matching metrics for inferring the class of a given query from the support set examples, following the usual approach for inference with prototypical networks. Experiments with four different adaptations of two main backbones demonstrate the effectiveness of our approach by achieving state-of-the-art results in miniImageNet, tieredImageNet, and CUB datasets. For fair comparison, the parameters of all the adapted backbones are reduced according to the number of parameters added by the mappers.



(a) input (b) baseline (c) ours

Figure 5. Comparision of gradient saliency maps. From left, we look at the (a) input original image, (b) baseline [8], and (c) subset of five feature vectors extracted by SetFeat12. The figure presents three examples of the training data in the first rows and four examples from the valid. set of miniImageNet in the last four rows.

Limitations Even though a comparison with different mapper configurations has been provided in sec. 6.1, we have evaluated our method using a fixed set of $M = 10$ mappers. Using more mappers ($M > 10$) has been considered, but was eventually dismissed since increasing the number of mappers would require reducing the number of filters, which in turn could cause underfitting due to underparameterization. As future work, we see great potential on analyzing the effect of increasing the number of mappers, possibly with larger backbones. Another topic requiring further investigations would be to vary the weighting of each mapper through more flexible set-to-set matching metrics. Although the min-sum and min-min metric non-linearly match the feature sets (through the $\min$ operation), investigating the weighted sum-min would be an interesting future work. Here, adapting Deep Set [71] before computing the min-sum metric would be a potential direction to investigate the weighted set-to-set mapping. Finally, we are particularly enthusiastic regarding the adaptation of our approach to self-supervised, since the set of features provide more choices for the comparison of different variations of single images.

Acknowledgements This project was supported by NSERC grant RGPIN-2020-04799, Mitacs, Prompt-Quebec, and Compute Canada. We thank A. Schwerdtfeger, A. Tupper, C. Shui, I. Hedhli, for proofreading the manuscript.

References

[1] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagné. Associative alignment for few-shot image classification. In *European Conference on Computer Vision*, 2020. 2

[2] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagné. Mixture-based feature space learning for few-shot image classification. *International Conference on Computer Vision*, 2021. 5, 6

[3] Kelsey R Allen, Evan Shelhamer, Hanul Shin, and Joshua B Tenenbaum. Infinite mixture prototypes for few-shot learning. *International Conference on Machine Learning, PMLR*, 2019. 5

[4] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *International Conference on Learning Representations*, 2019. 2

[5] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. 2

[6] Malik Boudiaf, Ziko Imtiaz Masud, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Transductive information maximization for few-shot learning. *Neural Information Processing Systems*, 2020. 2

[7] Qi Cai, Yingwei Pan, Ting Yao, Chenggang Yan, and Tao Mei. Memory matching networks for one-shot image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2018. 5

[8] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *International Conference on Learning Representations*, 2019. 2, 5, 6, 8

[9] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *International Conference on Computer Vision*, 2021. 5

[10] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[11] Wen-Hsuan Chu, Yu-Jhe Li, Jing-Cheng Chang, and Yu-Chiang Frank Wang. Spot and learn: A maximum-entropy patch sampler for few-shot image classification. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[12] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *International Conference on Learning Representations*, 2020. 2

[13] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *Neural Information Processing Systems*, 2021. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2020. 2, 3

[15] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *International Conference on Computer Vision*, 2019. 6

[16] Nanyi Fei, Zhiwu Lu, Tao Xiang, and Songfang Huang. Melr: Meta-learning via modeling episode-level relationships for few-shot learning. In *International Conference on Learning Representations*, 2021. 5, 6

[17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017. 2, 5, 6

[18] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Neural Information Processing Systems*, 2018. 5

[19] Hang Gao, Zheng Shou, Alireza Zareian, Hanwang Zhang, and Shih-Fu Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. In *Neural Information Processing Systems*, 2018. 2

[20] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *International Conference on Learning Representations*, 2018. 2

[21] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *International Conference on Computer Vision*, 2019. 2, 5

[22] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2, 5

[23] Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[24] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: A vision transformer in convnet's clothing for faster inference. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[25] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *International Conference on Computer Vision*, 2017. 2

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016. 3, 5

[27] Xiaopeng Hong, Hong Chang, Shiguang Shan, Xilin Chen, and Wen Gao. Sigma set: A small second order statistical region descriptor. In *Conference on Computer Vision and Pattern Recognition*, 2009. 2

[28] Ruibing Hou, Hong Chang, Bingpeng MA, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *Neural Information Processing Systems*, 2019. 2

[29] Junsik Kim, Tae-Hyun Oh, Seokju Lee, Fei Pan, and In So Kweon. Variational prototyping-encoder: One-shot learning with prototypical images. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[30] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, 2006. 2

[31] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Conference on Computer Vision and Pattern Recognition*, 2019. 5, 6

[32] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[33] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense classification and implanting for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[34] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2

[35] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *European Conference on Computer Vision*, 2020. 5, 6

[36] Bin Liu, Zhirong Wu, Han Hu, and Stephen Lin. Deep metric transfer for label propagation with limited annotated data. In *International Conference on Computer Vision Workshops*, 2019. 2

[37] Akshay Mehrotra and Ambedkar Dukkipati. Generative adversarial residual pairwise networks for one shot learning. *arXiv preprint arXiv:1703.08033*, 2017. 2

[38] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*, 2018. 5

[39] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. Boil: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021. 5

[40] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Neural Information Processing Systems*, 2018. 2, 5

[41] Hang Qi, Matthew Brown, and David G. Lowe. Low-shot learning with imprinted weights. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2

[42] Pedro Quelhas, Florent Monay, J-M Odobez, Daniel Gatica-Perez, Tinne Tuytelaars, and Luc Van Gool. Modeling scenes with local descriptors and latent aspects. In *International Conference on Computer Vision*, 2005. 2

[43] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2016. 2

[44] 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. *International Conference on Learning Representations*, 2018. 2, 5

[45] Mamshad Nayeem Rizve, Salman Khan, Fahad Shahbaz Khan, and Mubarak Shah. Exploring complementary strengths of invariant and equivariant representations for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2021. 2, 6

[46] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Neural Information Processing Systems*, 2018. 2

[47] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2020. 6

[48] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*. Citeseer, 2014. 8

[49] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Neural Information Processing Systems*, 2017. 1, 2, 4, 5, 6, 7

[50] Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. When does self-supervision improve few-shot learning? In *European Conference on Computer Vision*, 2020. 2

[51] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2019. 6

[52] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2, 5, 6

[53] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *Neural Information Processing Systems*, 2020. 6

[54] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Few-shot image classification: a good embedding is all you need. In *European Conference on Computer Vision*, 2020. 2, 5, 6

[55] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. *International Conference on Learning Representations*, 2020. 2

[56] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008. 7

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017. 2, 3

[58] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Neural Information Processing Systems*, 2016. 2, 3, 4, 5, 6

[59] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset, 2011. 5

[60] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2

[61] Yu-Xiong Wang and Martial Hebert. Learning from small sample sets by combining unsupervised meta-training with cnns. In *Neural Information Processing Systems*, 2016. 2

[62] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[63] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 2, 5

[64] Ziyang Wu, Yuwei Li, Lihua Guo, and Kui Jia. Parn: Position-aware relation networks for few-shot learning. In *International Conference on Computer Vision*, 2019. 5

[65] Chengming Xu, Yanwei Fu, Chen Liu, Chengjie Wang, Jilin Li, Feiyue Huang, Li Zhang, and Xiangyang Xue. Learning dynamic alignment via meta-filter for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2021. 2, 4, 5, 6

[66] Weijian Xu, Huaijin Wang, Zhuowen Tu, et al. Attentional constellation nets for few-shot learning. In *International Conference on Learning Representations*, 2020. 2

[67] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Conference on Computer Vision and Pattern Recognition*, 2020. 2, 4, 5, 6

[68] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Conference on Neural Information Processing Systems*, 2018. 2

[69] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. *International Conference on Machine Learning*, 2019. 2, 6

[70] Zhongjie Yu, Lin Chen, Zhongwei Cheng, and Jiebo Luo. Transmatch: A transfer-learning scheme for semi-supervised few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2020. 2

[71] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. In *Advances in neural information processing systems*, 2017. 2, 8

[72] Baoquan Zhang, Xutao Li, Yunming Ye, Zhichao Huang, and Lisai Zhang. Prototype completion with primitive knowledge for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2021. 2

[73] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Conference on Computer Vision and Pattern Recognition*, 2020. 2, 4, 5, 6

[74] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018. 2

[75] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. Few-shot learning via saliency-guided hallucination of samples. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[76] Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. Variational few-shot learning. In *International Conference on Computer Vision*, 2019. 2

[77] Manli Zhang, Jianhong Zhang, Zhiwu Lu, Tao Xiang, Mingyu Ding, and Songfang Huang. Iept: Instance-level and episode-level pretext tasks for few-shot learning. In *International Conference on Learning Representations*, 2020. 2