This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Finding Task-Relevant Features for Few-Shot Learning by Category Traversal

Hongyang Li[1,2*] David Eigen[2] Samuel Dodge[2] Matthew Zeiler[2] Xiaogang Wang[1,3]
[1]The Chinese University of Hong Kong [2]Clarifai Inc. [3]SenseTime Research
{yangli,xgwang}@ee.cuhk.edu.hk {deigen,samuel,zeiler}@clarifai.com

Abstract

Few-shot learning is an important area of research. Conceptually, humans are readily able to understand new concepts given just a few examples, while in more pragmatic terms, limited-example training situations are common in practice. Recent effective approaches to few-shot learning employ a metric-learning framework to learn a feature similarity comparison between a query (test) example, and the few support (training) examples. However, these approaches treat each support class independently from one another, never looking at the entire task as a whole. Because of this, they are constrained to use a single set of features for all possible test-time tasks, which hinders the ability to distinguish the most relevant dimensions for the task at hand. In this work, we introduce a Category Traversal Module *that can be inserted as a plug-and-play module into most metric-learning based few-shot learners. This component traverses across the* entire support set at once, *identifying task-relevant features based on both intra-class commonality and inter-class uniqueness in the feature space. Incorporating our module improves performance considerably (5%-10% relative) over baseline systems on both mini-ImageNet and tieredImageNet benchmarks, with overall performance competitive with recent state-of-the-art systems.*

1. Introduction

The goal of few-shot learning [38, 35, 30, 36, 33, 7, 25, 26] is to classify unseen data instances (*query* examples) into a set of new categories, given just a small number of labeled instances in each class (*support* examples). Typically, there are between 1 and 10 labeled examples per class in the support set; this stands in contrast to the standard classification problem [19, 16, 20], in which there are often thousands per class. Also classes for training and test set are the same in traditional problem whereas in few-shot learning the two sets are exclusive. A key challenge in few-shot learning, therefore, is to make best use of the limited data available in the support set in order to find the "right" generalizations as suggested by the task.

[*]This paper is the product of work during an internship at Clarifai Inc.

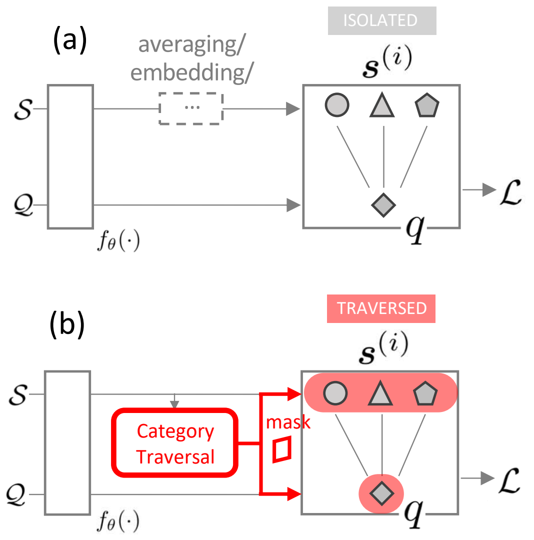

Figure 1. **(a)** A high-level illustration of the metric-based algorithms for few-shot learning. Both support and query are first fed into a feature extractor f_θ; in previous methods, the query is compared with support based on the feature similarity *individually*, without associating the most relevant information across classes. **(b)** The proposed Category Traversal Module (CTM) looks at *all* categories in the support set to find task-relevant features.

A recent effective approach to this problem is to train a neural network to produce feature embeddings used to compare the query to each of the support samples. This is similar to metric learning using Siamese Networks [4, 6], trained explicitly for the few-shot classification problem by iteratively sampling a query and support set from a larger labeled training set, and optimizing to maximize a similarity score between the query and same-labeled examples. Optimizing for similarity between query and support has been very effective [35, 38, 31, 14, 21, 2, 13]. Fig. 1 (a) illustrates such a mechanism at a high level.

However, while these approaches are able to learn rich features, the features are generated for each class in the sup-

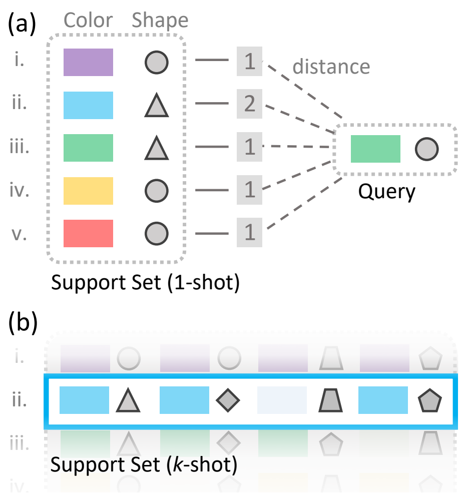

Figure 2. Toy example illustrating the motivation for task-relevant features. **(a)** A task defines five classes (i)..(v) with two feature dimensions, color and shape. The distance between query and support is the same for classes (i, iii, iv, v). However, by taking the context of all classes into account, we see the relevant feature is color, so the class of the query image should be that of (iii): green. **(b)** In a k-shot ($k > 1$) case, most instances in one class share the property of color blue whilst their shape differ among them — making the color feature more representative.

port set independently. In this paper, we extend the effective metric-learning based approaches to incorporate the context of the *entire* support set, viewed as a whole. By including such a view, our model finds the dimensions most relevant to each task. This is particularly important in few-shot learning: since very little labeled data is available for each task, it is imperative to make best use of all of the information available from the full set of examples, taken together.

As a motivating example, consider Fig.2 (a), which depicts a 5-way 1-shot task with two simple feature dimensions, color and shape. The goal of classification is to determine the answer to a question:

To which category in the support set does query belong?

Here, the query is a green circle. We argue that because each example in the support set has a unique color, but shares its shape with other support examples, the relevant feature in this task is color. Therefore, the correct label assignment to the query should be class (iii): green. However, if feature similarity to the query is calculated independently for each class, as is done in [38, 35, 36, 34, 21, 30, 23, 10], it is impossible to know which dimension (color or shape) is more relevant, hence categories (i, iii, iv, v) would all have equal score based on distance[1]. Only by looking at context of the entire support set can we determine that color is the discriminating dimension for this task. Such an observation motivates us to traverse all support classes to find the *inter-class uniqueness* along the feature dimension.

[1]Note that if the feature extractor were to learn that color is more important than shape in order to succeed in this particular task, it would fail on the task where color is shared and shape is unique — it is impossible for static feature comparison to succeed in both tasks.

Moreover, under the multi-shot setting shown in Fig. 2 (b), it can be even clearer that the color dimension is most relevant, since most instances have the same color of blue, while their shape varies. Thus, in addition to inter-class uniqueness, relevant dimensions can also be found using the *intra-class commonality*. Note in this $k > 1$ case, feature averaging within each class is an effective way to reduce intra-class variations and expose shared feature components; this averaging is performed in [35, 36]. While both inter- and intra-class comparisons are fundamental to classification, and have long been used in machine learning and statistics [8], metric based few-shot learning methods have not yet incorporated any of the context available by looking between classes.

To incorporate both inter-class as well as intra-class views of support set, we introduce a **category traversal module** (CTM). Such a module selects the most relevant feature dimensions after traversing both across and within categories. The output of CTM is bundled onto the feature embeddings of the support and query set, making metric learning in the subsequent feature space more effective. CTM consists of a concentrator unit to extract the embeddings within a category for commonality, and a projector unit to consider the output of the concentrator across categories for uniqueness. The concentrator and projector can be implemented as convolutional layer(s). Fig. 1 (b) gives a description of how CTM is applied into existing metric-based few-shot learning algorithms. It can be viewed as a plug-and-play module to provide more discriminative and representative features by considering the global feature distribution in the support set – making metric learning in high-dimensional space more effective.

We demonstrate the effectiveness of our category traversal module on the few-shot learning benchmarks. CTM is on par with or exceeding previous state-of-the-art. Incorporating CTM into existing algorithms [36, 38, 35], we witness consistent relative gains of around 5%-10% on both *mini*ImageNet and *tiered*ImageNet. The code suite is at: `https://github.com/Clarifai/few-shot-ctm`.

2. Related Work

Recent years have witnessed a vast amount of work on the few-shot learning task. They can be roughly categorized into three branches, (i) metric based, (ii) optimization based, and (iii) large corpus based.

The first branch of works are **metric based** approaches [35, 38, 36, 14, 34, 21, 30, 23, 2, 10]. Vinyals *et al.* [38] introduced the concept of episode training in few-shot learning, where the training procedure mimics the test scenario based on support-query metric learning. The idea is intuitive and simple: these methods compare feature similarity after embedding both support and query samples into a shared feature space. The prototypical network [35] is built upon [38] by comparing the query with class prototypes in the support set, making use of class centroids to eliminate the outliers in the support set and find dimensions common to all class samples. Such a practice is similar in spirit to our concentrator module, which we design to focus on intra-class commonality. Our work goes beyond this by also looking at all classes in the support set together to find dimensions relevant for each task. In [14], a kernel generator is introduced to modify feature embeddings, conditioned on the query image. This is a complementary approach to ours: while [14] looks to the query to determine what may be relevant for its classification, we look at the whole of the support set to enable our network to better determine which features most pertain to the task. In [34], the feature embedding and classifier weight creation networks are broken up, to enable zero-shot and few-shot tasks to both be performed within the same framework.

There are also interesting works that explore the relationship between support and query to enable more complex comparisons between support and query features. The relation network [36] proposes evaluating the relationship of each query-support pair using a neural network with concatenated feature embeddings. It can be viewed as a further extension to [35, 38] with a learned metric defined by a neural network. Liu *et al.* [23] propose a transductive propagation network to propagate labels from known labeled instances to unlabeled test instances, by learning a graph construction module that exploits the manifold structure in the data. Garcia *et al.* [10] introduced the concept of a graph neural network to explicitly learn feature embeddings by assimilating message-passing inference algorithms into neural-network counterparts. Oreshkin *et al.* [26] also learns a task-dependent metric, but conditions based on the mean of class prototypes, which can reduce inter-class variations available to their task conditioning network, and requires an auxiliary task co-training loss not needed by our method to realize performance gains. Gao *et al.* [9] applied masks to features in a prototypical network applied to a NLP few-shot sentence classification task, but base their masks only on examples within each class, *not* between classes as our method does.

All the approaches mentioned above base their algorithms on a metric learning framework that compares the query to each support class, taken separately. However, none of them incorporate information available *across* categories for the task, beyond the final comparison of individually-created distance scores. This can lead to problems mentioned in Section 1, where feature dimensions irrelevant to the current task can end up dominating the similarity comparison. In this work, we extend metric-based approaches by introducing a category traversal module to find relevant feature dimensions for the task by looking at all categories simultaneously.

The second branch of literature are **optimization based** solutions [28, 22, 7, 25, 33]. For each task (episode), the learner samples from a distribution and performs SGD or unrolled weight updates for a few iterations to adapt a parameterized model for the particular task at hand. In [28], a learner model is adapted to a new episodic task by a recurrent meta-learner producing efficient parameter updates. MAML [7] and its variants [25, 33] have demonstrated impressive results; in these works, the parameters of a learner model are optimized so that they can be quickly adapted to a particular task.

At a high-level, these approaches incorporate the idea of traversing all support classes, by performing a few weight update iterations for the few-shot task. However, as pointed out by [33, 21], while these approaches iterate over samples from all classes in their task updates, they often have trouble learning effective embeddings. [33] address this by applying the weight update "inner-loop" only to top layer weights, which are initialized by sampling from a generative distribution conditioned on the task samples, and pretraining visual features using an initial supervised phase. By contrast, metric learning based methods achieve considerable success in learning good features, but have ***not*** made use of inter-class views to determine the most relevant dimensions for each task. We incorporate an ***all-class view*** into a metric learning framework, and obtain competitive performance. Our proposed method learns both the feature embeddings and classification dimensions, and is trained in an entirely from-scratch manner.

The third branch is **large-training-corpus based** methods [11, 15, 12, 27, 29]. In these, a base network is trained with large amount of data, but also must be able to adapt to a few-shot learning task without forgetting the original base model concepts. These methods provide stronger feature representations for base model classes that are still "compatible" with novel few-class concept representations, so that novel classes with few examples can be readily mixed with classes from the base classifier.

3. Algorithm

3.1. Description on Few-Shot Learning

In a few-shot classification task, we are given a small support set of N distinct, previously unseen classes, with K examples each[2]. Given a query sample, the goal is to classify it into one of the N support categories.

Training. The model is trained using a large training corpus $\mathbb{C}^{\texttt{train}}$ of labeled examples (of categories different

[2]Typically, N is between 5 and 20, and K between 1 and 20.

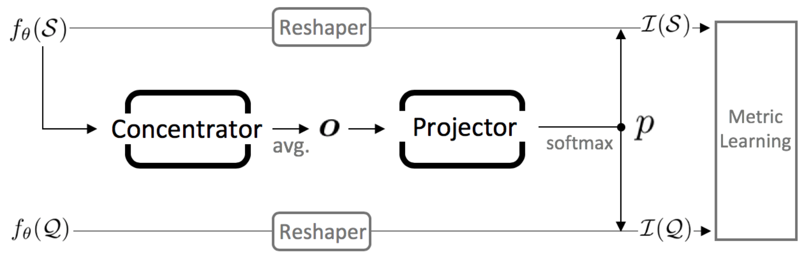

Figure 3. Detailed breakdown of components in CTM. It extracts features common to elements in each class via a concentrator $\boldsymbol{o}$, and allows the metric learner to concentrate on more discriminative dimensions via a projector p, constructed by traversing all categories in the support set.

from any that we will see in the eventual few-shot tasks during evaluation). The model is trained using *episodes*. In each episode, we construct a support set $\mathcal{S}$ and query set $\mathcal{Q}$:

$$\mathcal{S} = \{\boldsymbol{s}^{(1)}, \cdots, \boldsymbol{s}^{(c)}, \cdots, \boldsymbol{s}^{(N)}\} \subset \mathbb{C}^{\texttt{train}}, |\boldsymbol{s}^{(c)}| = K,$$
$$\mathcal{Q} = \{\boldsymbol{q}^{(1)}, \cdots, \boldsymbol{q}^{(c)}, \cdots, \boldsymbol{q}^{(N)}\} \subset \mathbb{C}^{\texttt{train}}, |\boldsymbol{q}^{(c)}| = K,$$

where c is the class index and K is the number of samples in class $\boldsymbol{s}^{(c)}$; the support set has a total number of NK samples and corresponds to a N-way K-shot problem. Let s_j be a single sample, where j is the index among all samples in $\mathcal{S}$. We define the label of sample i to be:

$$l_i^* \triangleq l(s_i) = c, \quad s_i \in \boldsymbol{s}^{(c)}.$$

Similar notation applies for the query set $\mathcal{Q}$.

As illustrated in Fig. 1, the samples s_i, q_j are first fed into a feature extractor $f_\theta(\cdot)$. We use a CNN or ResNet [16] as the backbone for f_θ. These features are used as input to a comparison module $\mathcal{M}(\cdot,\cdot)$. In practice, $\mathcal{M}$ could be a direct pair-wise feature distance [35, 38] or a further relation unit [36, 10] consisting additional CNN layers to measure the relationship between two samples. Denote the output score from $\mathcal{M}$ as $Y = \{y_{ij}\}$. The loss $\mathcal{L}$ for this training episode is defined to be a cross-entropy classification loss averaged across all query-support pairs:

$$y_{ij} = \mathcal{M}\big(f_\theta(s_i), f_\theta(q_j)\big), \tag{1}$$
$$\mathcal{L} = -\frac{1}{(NK)^2}\sum_i\sum_j \mathbf{1}[l_i^* = l_j^*]\log y_{ij}. \tag{2}$$

Training proceeds by iteratively sampling episodes, and performing SGD update using the loss for each episode.

Inference. Generalization performance is measured on test set episodes, where $\mathcal{S}, \mathcal{Q}$ are now sampled from a corpus $\mathbb{C}^{\texttt{test}}$ containing classes distinct from those used in $\mathbb{C}^{\texttt{train}}$. Labels in the support set are known whereas those in the query are unknown, and used only for evaluation. The label prediction for the query is found by taking class with highest comparison score:

$$\hat{l}_j = \arg\max_c y_{cj}, \tag{3}$$

where $y_{cj} = \frac{1}{K}\sum_i y_{ij}$ and $l_i^* = c$. The mean accuracy is therefore obtained by comparing $\hat{l}_j$ with query labels for a length of test episodes (usually 600).

3.2. Category Traversal Module (CTM)

Fig. 3 shows the overall design of our model. The category traversal module takes support set features $f_\theta(\mathcal{S})$ as input, and produces a mask p via a concentrator and projector that make use of intra- and inter-class views, respectively. The mask p is applied to reduced-dimension features of both the support and query, producing improved features $\mathcal{I}$ with dimensions relevant for the current task. These improved feature embeddings are finally fed into a metric learner.

3.2.1 Concentrator: Intra-class Commonality

The first component in CTM is a *concentrator* to find universal features shared by all instances for one class. Denote the output shape from feature extractor f_θ as (NK, m_1, d_1, d_1), where m_1, d_1 indicate the number of channel and the spatial size, respectively. We define the concentrator as follows:

$$f_\theta(\mathcal{S}) : (NK, m_1, d_1, d_1) \xrightarrow{\text{Concentrator}} \boldsymbol{o} : (N, m_2, d_2, d_2), \tag{4}$$

where m_2, d_2 denote the output number of channel and spatial size. Note that the input is first fed to a CNN module to perform dimension reduction; then samples within each class are averaged to have the final output $\boldsymbol{o}$. In the 1-shot setting, there is no average operation, as there is only one example for each class.

In practice the CNN module could be either a simple CNN layer or a ResNet block [16]. The purpose is to remove the difference among instances and extract the commonality among instances within one category. This is achieved by way of an appropriate down-sampling from m_1, d_1 to m_2, d_2. Such a learned component is proved to be better than the averaging alternative [35], where the latter could be deemed as a special case of our concentrator when $m_1 = m_2, d_1 = d_2$ without the learned parameters.

3.2.2 Projector: Inter-class Uniqueness

The second component is a *projector* to mask out irrelevant features and select the ones most discriminative *for the current few-shot task* by looking at concentrator features from

Model	5-way 1-shot	5-way 5-shot	model size (Mb)	training time (sec. / episode)	20-way 1-shot	20-way 5-shot
(i) sample-wise style baseline	37.20%	53.35%	0.47	0.0545	17.96%	28.47%
(ii) sample-wise, $\mathcal{I}^1$	**41.62%**	**58.77%**	0.55	0.0688	**21.75%**	**32.26%**
(iii) baseline_same_size	37.82%	53.46%	0.54	0.0561	18.11%	28.54%
(iv) cluster-wise style baseline	34.81%	50.93%	0.47	0.0531	16.95%	27.37%
(v) cluster-wise, $\mathcal{I}^2$	39.55%	56.95%	0.55	0.0632	19.96%	30.17%

Table 1. Design choice of $\mathcal{I}(\mathcal{S})$ in the category traversal module (CTM) and comparison with baselines. We see a substantial improvement using CTM over the same-capacity baseline (ii, iii). The sample wise choice (ii) performs better, with marginal extra computation cost compared with (v).

all support categories simultaneously:

$$\hat{\boldsymbol{o}} : (1, Nm_2, d_2, d_2) \xrightarrow{\text{Projector}} p : (1, m_3, d_3, d_3). \qquad (5)$$

where $\hat{\boldsymbol{o}}$ is just a reshaped version of $\boldsymbol{o}$; m_3, d_3 follow similar meaning as in the concentrator. We achieve the goal of traversing across classes by concatenating the class prototypes in the first dimension (N) to the channel dimension (m_2), applying a small CNN to the concatenated features to produce a map of size $(1, m_3, d_3, d_3)$, and finally applying a softmax over the channel dimension m_3 (applied separately for each of the $d_3 \times d_3$ spatial dimensions) to produce a mask p. This is used to mask the relevant feature dimensions for the task in the query and support set.

3.2.3 Reshaper

In order to make the projector output p influence the feature embeddings $f_\theta(\cdot)$, we need to match the shape between these modules in the network. This is achieved using a reshaper network, applied separately to each of NK samples:

$$f_\theta(\cdot) \xrightarrow{\text{Reshaper}} \boldsymbol{r}(\cdot) : (NK, m_3, d_3, d_3).$$

It is designed in light-weight manner with one CNN layer.

3.2.4 Design choice in CTM

Armed by the components stated above we can generate a mask output by traversing all categories: $f_\theta(\mathcal{S}) \rightarrow p$. The effect of CTM is achieved by bundling the projector output onto the feature embeddings of both support and query, denoted as $\mathcal{I}(\cdot)$. The improved feature representations are thus promised to be more discriminative to be distinguished.

For the query, the choice of $\mathcal{I}$ is simple since we do not have labels of query; the combination is an element-wise multiplication of embeddings and the projector output: $\mathcal{I}(\mathcal{Q}) = \boldsymbol{r}(\mathcal{Q}) \odot p$, where $\odot$ stands for broadcasting the value of p along the sample dimension (NK) in $\mathcal{Q}$.

For the support, however, since we know the query labels, we can choose to mask p directly onto the embeddings (sample-wise), or if we keep $(m_2, d_2, d_2) = (m_3, d_3, d_3)$, we can use it to mask the concentrator output $\boldsymbol{o}$ (cluster-wise). Mathematically, these two options are:

$$\text{option 1: } \mathcal{I}^1(\mathcal{S}) = \boldsymbol{r}(\mathcal{S}) \odot p : (NK, m_3, d_3, d_3),$$
$$\text{option 2: } \mathcal{I}^2(\mathcal{S}) = \boldsymbol{o} \odot p : (N, m_3, d_3, d_3).$$

We found that option 1 results in better performance, for a marginal increase in execution time due to its larger number of comparisons; details are provided in Sec. 4.2.1.

3.3. CTM in Action

The proposed category traversal module is a simple plug-and-play module and can be embedded into any metric-based few-shot learning approach. In this paper, we consider three metric-based methods and apply CTM to them, namely the matching network [38], the prototypical network [35] and the relation network [36]. As discussed in Sec. 1, all these three methods are limited by not considering the entire support set simultaneously. Since features are created independently for each class, embeddings irrelevant to the current task can end up dominating the metric comparison. These existing methods define their similarity metric following Eqn. (1); we modify them to use our CTM as follows:

$$Y = \mathcal{M}\big(\boldsymbol{r}(\mathcal{S}) \odot p, \boldsymbol{r}(\mathcal{Q}) \odot p\big), \quad Y = \{y_{ij}\}. \qquad (6)$$

As we show later (see Sec. 4.3.1), after integrating the proposed CTM unit, these methods get improved by a large margin (2%-4%) under different settings.

4. Evaluation

The experiments are designed to answer the following key questions: (1) Is CTM competitive to other state-of-the-art on large-scale few-shot learning benchmarks? (2) Can CTM be utilized as a simple plug-and-play and bring in gain to existing methods? What are the essential components and factors to make CTM work? (3) How does CTM modify the feature space to make features more discriminative and representative?

4.1. Datasets and Setup

Datasets. The *mini*ImageNet dataset [38] is a subset of 100 classes selected from the ILSVRC-12 dataset [32] with 600 images in each class. It is divided into training, validation, and test meta-sets, with 64, 16, and 20 classes respectively. The *tiered*ImageNet dataset [30] is a larger subset of ILSVRC-12 with 608 classes (779,165 images) grouped into 34 higher-level nodes based on WordNet hierarchy [5]. This set of nodes is partitioned into 20, 6, and 8 disjoint sets of training, validation, and testing nodes, and the corresponding classes consist of the respective meta-sets. As argued in [30], the split in *tiered*ImageNet is more challenging, with realistic regime of test classes that are less similar to training ones. Note that the validation set is only used for tuning model parameters.

Evaluation metric. We report the mean accuracy (%) of 600 randomly generated episodes as well as the 95% confidence intervals on test set. In every episode during test, each class has 15 queries, following most methods [35, 36, 33].

Implementation details. For training, the 5-way problem has 15 query images while the 20-way problem has 8 query images. The reason for a fewer number of query samples in the 20-way setting is mainly due to the GPU memory considerations. The input image is resized to 84×84.

We use Adam [18] optimizer with an initial learning rate of 0.001. The total training episodes on *mini*ImageNet and *tiered*ImageNet are 600,000 and 1,000,000 respectively. The learning rate is dropped by 10% every 200,000 episodes or when loss enters a plateau. The weight decay is set to be 0.0005. Gradient clipping is also applied.

4.2. Ablation Study

4.2.1 Shallow Network Verification

We first validate the effectiveness of category traversal by comparing against same-capacity baselines using a simple backbone network. Specifically, a 4-layer neural network is adopted as backbone; we directly compute feature similarity between $\mathcal{I}(\mathcal{S})$ and $\mathcal{I}(\mathcal{Q})$. The mean accuracy on *mini*ImageNet is reported. After feature embedding, $m_1 = 64, d_1 = 21$; the concentrator is a CNN layer with stride of 2, *i.e.*, $m_2 = 32, d_2 = 10$. To compare between choices ($\mathcal{I}^1$ or $\mathcal{I}^2$), the projector leaves dimensions unchanged, *i.e.*, $m_3 = m_2, d_3 = d_2$.

Baseline comparison. Results are reported in Tab. 1. Model size and training time are measured under the 5-way 5-shot setting. The "baseline" in row (i) and (iv) evaluate a model with reshaper network and metric comparisons only, omitting CTM concentrator and projector. Row (ii) shows a model that includes our CTM. Since adding CTM increases the model capacity compared to the baseline (i), we also include a same-size model baseline for comparison, shown as "baseline_same_size" (iii), by adding additional layers to the backbone such that its model size is similar to (ii). Note that the only difference between (i) and (iv) is that the latter case takes average of samples within each category.

We can see on average there is a 10% relative improvement using CTM in both 5-way and 20-way settings, compared to the baselines. Notably, the larger-capacity baseline improves only marginally over the original baseline, while the improvements using CTM are substantial. This shows that the performance increase obtained by CTM is indeed due to its ability to find relevant features for each task.

Table 2. Ablation study on category traversal module.

Factor	***mini*ImageNet accuracy**	
	1-shot	5-shot
CTM with shallow (4 layer) backbone	41.62	58.77
CTM with ResNet-18 backbone	**59.34**	**77.95**
(i) w/o concentrator network $\boldsymbol{o}$	55.41	73.29
(ii) w/o projector p	57.18	74.25
(iii) softmax all in p	57.77	75.03
relation net baseline without CTM	58.21	74.29
relation net $\mathcal{M}$, CTM, MSE loss	61.37	78.54
relation net $\mathcal{M}$, CTM, cross entropy loss	**62.05**	**78.63**

Which option for $\mathcal{I}(\mathcal{S})$ is better? Table 1 (ii, v) shows the comparison between $\mathcal{I}^1$ and $\mathcal{I}^2$. In general, the sample-wise choice $\mathcal{I}^1$ is 2% better than $\mathcal{I}^2$. Note the model size between these two are exactly the same; the only difference is how p is multiplied. However, a trivial drawback of $\mathcal{I}^1$ is the slightly slower time (0.0688 vs 0.0632) since it needs to broadcast p across all samples. Despite the efficiency, we choose the first option as our preference to generate $\mathcal{I}(\mathcal{S}) = \mathcal{I}^1$ nonetheless.

4.2.2 CTM with Deeper Network

Table 2 reports the ablation analysis on different components of CTM. Using a deeper backbone for the feature extractor increases performance by a large margin. Experiments in the second block investigate the effect of the concentrator and projector, respectively. Removing each component alone results in a performance decrease (cases i, ii, iii)[3]. The accuracy is inferior (-3.93%, 1-shot case) if we remove the network part of the concentrator, implying that its dimension reduction and spatial downsampling is important to the final comparisons. Removing the projector p also results in a significant drop (-2.16%, 1-shot), confirming that this step is necessary to find task-specific discriminate dimensions. An interesting result is that if we perform the softmax operation across all the locations (m_3, d_3, d_3) in p, the accuracy (57.77%) is inferior to performing softmax along the channel dimension (m_3) for each location separately (59.34%); this is consistent with the data, where absolute position in the image is only modestly relevant to any class difference.

Moreover, we incorporate the relation module [36] as the metric learner for the last module $\mathcal{M}$. It consists of two CNN blocks with two subsequent fc layers generating the relationship score for one query-support pair. The baseline relation net model without CTM has an accuracy of 58.21%. After including our proposed module, the performance in-

[3]Implementation details: case (i) without concentrator, support samples are still averaged to generate an output of (N, m, d, d) for the projector; case (ii) without projector, the improved feature representation for support and query are $\boldsymbol{o}(\mathcal{S}), \boldsymbol{r}(\mathcal{Q})$, respectively.

Method	5-way		20-way		5-way		20-way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Matching Net [38], *paper*	43.56	55.31	-	-	-	-	-	-
Matching Net [38], *our implementation*	48.89	66.35	23.18	36.73	54.02	70.11	23.46	41.65
Matching Net [38], *CTM*	**52.43**	**70.09**	**25.84**	**40.98**	**57.01**	**73.45**	**25.69**	**45.07**
	+3.54	*+3.74*	*+2.66*	*+4.25*	*+2.99*	*+3.34*	*+2.23*	*+3.42*
Prototypical Net [35], *paper*	49.42	68.20	-	-	53.31	72.69	-	-
Prototypical Net [35], *our implementation*	56.11	74.16	28.53	42.36	60.27	75.80	28.56	49.34
Prototypical Net [35], *CTM*	**59.34**	**77.95**	**32.08**	**47.11**	**63.77**	**79.24**	**31.02**	**51.44**
	+3.23	*+3.79*	*+3.55*	*+4.75*	*+3.50*	*+3.44*	*+2.46*	*+2.10*
Relation Net [36], *paper*	50.44	65.32	-	-	54.48	71.32	-	-
Relation Net [36], *our implementation*	58.21	74.29	31.35	45.19	61.11	77.39	26.77	47.82
Relation Net [36], *CTM*	**62.05**	**78.63**	**35.11**	**48.72**	**64.78**	**81.05**	**31.53**	**52.18**
	+3.84	*+4.34*	*+3.76*	*+3.53*	*+3.67*	*+3.66*	*+4.76*	*+4.36*

Table 3. Improvement after incorporating CTM into existing methods on *mini*ImageNet (left) and *tiered*ImageNet (right).

creases by 3.84%, to 62.05%. Note that the original paper [36] uses mean squared error (MSE); we find cross-entropy is slightly better (0.68% and 0.09% for 1-shot and 5-shot, respectively), as defined in Eqn. (2).

4.3. Comparison with State-of-the-Art

4.3.1 Adapting CTM into Existing Frameworks

To verify the effectiveness of our proposed category traversal module, we embed it into three metric-based algorithms that are closely related to ours. It is worth noticing that the comparison should be conducted in a fair setting; however, different sources report different results[4]. Here we describe the implementation we use.

Matching Net [38] and Prototypical Net [35]. In these cases, the metric module $\mathcal{M}$ is the pair-wise feature distance. Note that a main source of improvement between [38] and [35] is that the query is compared to the average feature for each class; this has the effect of including intra-class commonality, which we make use of in our concentrator module. As for the improvement from original paper to our baseline, we use the ResNet-18 model with a Euclidean distance for the similarity comparison, instead of a shallow CNN network with cosine distance originally.

Relation Net [36]. As for the improvement from original paper to our baseline, the backbone structure is switched from 4-conv to ResNet-18 model; the relation unit $\mathcal{M}$ adopts the ResNet blocks instead of CNN layers; the supervision loss is changed to the cross entropy.

Table 3 shows the gains obtained by including CTM into each method. We observe that on average, there is an approximately 3% increase after adopting CTM. This shows the ability of our module to plug-and-play into multiple metric based systems. Moreover, the gains remain consistent for each method, regardless of the starting performance level. This supports the hypothesis that our method is able to incorporate signals previously unavailable to any of these approaches, *i.e.,* the inter-class relations in each task.

[4]For example, the relation network has a 65.32% accuracy for 5-way 5-shot setting on *mini*ImageNet. [39] gives a 61.1%; [2] has 66.6%; [21] obtains 71.07% with a larger network

Model	***mini*ImageNet test accuracy**	
	1-shot	5-shot
Meta-learner LSTM [28]	43.44 ± 0.77	60.60 ± 0.71
MAML [7]	48.70 ± 1.84	63.11 ± 0.92
REPTILE [25]	49.97 ± 0.32	65.99 ± 0.58
Meta-SGD [22]	54.24 ± 0.03	70.86 ± 0.04
SNAIL [24]	55.71 ± 0.99	68.88 ± 0.92
CAML [17]	59.23 ± 0.99	72.35 ± 0.18
LEO [33]	61.76 ± 0.08	77.59 ± 0.12
Incremental [29]	55.72 ± 0.41	70.50 ± 0.36
Dynamic [12]	56.20 ± 0.86	73.00 ± 0.64
Predict Params [27]	59.60 ± 0.41	73.74 ± 0.19
Matching Net [38]	43.56 ± 0.84	55.31 ± 0.73
BANDE [1]	48.90 ± 0.70	68.30 ± 0.60
Prototypical Net [35]	49.42 ±0.78	68.20 ± 0.66
Relation Net [36]	50.44 ± 0.82	65.32 ± 0.70
Projective Subspace [34]	——	68.12 ± 0.67
Individual Feature [13]	56.89 ——	70.51 ——
IDeMe-Net [3]	57.71 ——	74.34 ——
TADAM [26]	58.50 ± 0.30	76.70 ± 0.30
CTM (ours)	**62.05** ± 0.55	**78.63** ± 0.06
CTM (ours), data augment	**64.12** ± 0.82	**80.51** ± 0.13

Model	***tiered*ImageNet test accuracy**	
	1-shot	5-shot
MAML [7]	51.67 ± 1.81	70.30 ± 0.08
Meta-SGD [22], reported by [33]	62.95 ± 0.03	79.34 ± 0.06
LEO [33]	**66.33** ± 0.05	**81.44** ± 0.09
Dynamic [12], reported by [29]	50.90 ± 0.46	66.69 ± 0.36
Incremental [29]	51.12 ± 0.45	66.40 ± 0.36
Soft k-means [30]	52.39 ± 0.44	69.88 ± 0.20
Prototypical Net [35]	53.31 ± 0.89	72.69 ± 0.74
Projective Subspace [34]	——	71.15 ± 0.67
Relation Net [36]	54.48 ± 0.93	71.32 ± 0.78
Transductive Prop. [23]	59.91 ——	73.30 ——
CTM (ours)	64.78 ± 0.11	81.05 ± 0.52
CTM (ours), data augment	**68.41** ± 0.39	**84.28** ± 1.73

Table 4. Test accuracies for 5-way tasks, both 1-shot and 5-shot. We provide two versions of our model. See Sec. 4.3.2 for details.

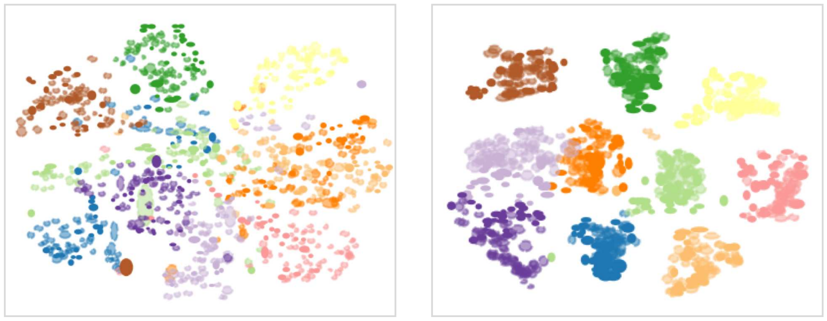
(a) Relation net, 47.82% accuracy (b) Relation Net with CTM, 52.18% accuracy

Figure 4. The t-SNE visualization [37] of the improved feature embeddings $\mathcal{I}(\cdot)$ learned by our CTM approach. (a) corresponds to the 20-way 5-shot setting of the relation network without CTM in Table 3 and (b) corresponds to the improved version with CTM. Only 10 classes are shown for better view. We can see that after traversing across categories, the effect of projector p onto the features are obvious - making clusters more compact and discriminative from each other.

4.3.2 Comparison beyond Metric-based Approaches

We compare our proposed CTM approach with other state-of-the-art methods in Table 4. For each dataset, the first block of methods are optimization-based, the second are base-class-corpus algorithms, and the third are metric-based approaches. We use a ResNet-18 backbone for the feature extractor to compare with other approaches. The model is trained from scratch with standard initialization, and no additional training data (*e.g.*, distractors [30, 23]) are utilized. We believe such a design aligns with most of the compared algorithms in a fair spirit.

It is observed that our CTM method compares favorably against most methods by a large margin, not limited to the metric-based methods but also compared with the optimization-based methods. For example, under the 5-way 1-shot setting, the performance is 62.05% vs 59.60% [27], and 64.78% vs 59.91% [23] on the two benchmarks *mini*ImageNet and *tiered*ImageNet, respectively.

LEO [33] is slightly better than ours (without data augmentation) on *tiered*ImageNet. It uses wide residual networks [40] with 28 layers; they also pretrain the model using a supervised task on the entire training set and finetune the network based on these pre-trained features. For practical interest, we also train a version of our model with supervised pretraining (using only the *mini*- or *tiered*ImageNet training sets), basic data augmentation (including random crop, color jittering and horizontal flip), and a higher weight decay (0.005). The result is shown in the last case for each dataset. Note that the network structure is *still* ResNet-18, considering LEO's wideResNet-28.

4.4. Feature Visualization Learned by CTM

Fig. 4 visualizes the feature distribution using t-SNE [37]. The features computed in a 20-way 5-shot setting, but only 10 classes are displayed for easier comparison. Model (a) achieves an accuracy of 47.32% without CTM and the improved version, Model (b), equipped with CTM has a better performance of 52.18%. When sampling features for t-SNE for our model, we use $\mathcal{I}(\mathcal{S})$, *i.e.* after the mask p is applied. Since this depends on the support sample, features will be vastly different depending on the chosen task. Therefore, when sampling tasks to create these visualization features, we first chose 20 classes, and kept these fixed while drawing different random support samples from this class set. We draw a total of 50 episodes on the test set.

As can be clearly observed, CTM model has more compact and separable clusters, indicating that features are more discriminative for the task. This descends from the design of the category traversal module. Without CTM, some clusters overlap with each other (*e.g.*, light green with orange), making the metric learning difficult to compare.

5. Conclusion

In this paper, we propose a category traversal module (CTM) to extract feature dimensions most relevant to each task, by looking the context of the entire support set. By doing so, it is able to make use of both inter-class uniqueness and intra-class commonality properties, both of which are fundamental to classification. By looking at all support classes together, our method is able to identify discriminative feature dimensions for each task, while still learning effective comparison features entirely from scratch. We devise a concentrator to first extract the feature commonality among instances within the class by effectively down-sampling the input features and averaging. A projector is introduced to traverse feature dimensions across all categories in the support set. The projector inter-class relations to focus on the on relevant feature dimensions for the task at hand. The output of CTM is then combined onto the feature embeddings for both support and query; the enhanced feature representations are more unique and discriminative for the task. We have demonstrated that it improves upon previous methods by a large margin, and has highly competitive performance compared with state-of-the-art.

Acknowledgment

We thank Nand Dalal, Michael Gormish, Yanan Jian and reviewers for helpful discussions and comments. H. Li is supported by Hong Kong Ph.D. Fellowship Scheme.

References

[1] Kelsey R Allen, Hanul Shin, Evan Shelhamer, and Josh B. Tenenbaum. Variadic learning by bayesian nonparametric deep embedding. In *OpenReview*, 2019.

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

[3] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In *OpenReview*, 2019.

[4] Sumit Chopra, Raia Hadsell, and Yann Lecun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[6] Nir Ailon Elad Hoffer. Deep Metric Learning using Triplet Network. In *ICLR*, 2015.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *arXiv preprint:1703.03400*, 2017.

[8] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[9] Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. Hybrid Attention-Based Prototypical Networks for Noisy Few-Shot Relation Classification . In *AAAI*, 2019.

[10] Victor Garcia and Joan Bruna. Few-shot Learning with Graph Neural Networks. In *ICLR*, 2018.

[11] Mohammad Ghasemzadeh, Fang Lin, Bita Darvish Rouhani, Farinaz Koushanfar, and Ke Huang. Agilenet: Lightweight dictionary-based few-shot learning. In *arXiv preprint:1805.08311*, 2018.

[12] Spyros Gidaris and Nikos Komodakis. Dynamic Few-Shot Visual Learning without Forgetting. In *CVPR*, 2018.

[13] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

[14] Chunrui Han, Shiguang Shan, Meina Kan, Shuzhe Wu, and Xilin Chen. Meta-learning with individualized feature space for few-shot classification. 2019.

[15] Bharath Hariharan and Ross Girshick. Low-shot Visual Recognition by Shrinking and Hallucinating Features. In *ICCV*, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[17] Xiang Jiang, Mohammad Havaei, Farshid Varno, Gabriel Chartrand, Nicolas Chapados, and Stan Matwin. Learning to learn with conditional class dependencies. In *ICLR*, 2019.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[20] Hongyang Li, Xiaoyang Guo, Bo Dai, Wanli Ouyang, and Xiaogang Wang. Neural network encapsulation. In *ECCV*, 2018.

[21] Kai Li, Martin Renqiang Min, Bing Bai, Yun Fu, and Hans Peter Graf. Network reparameterization for unseen class categorization. In *OpenReview*, 2019.

[22] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to Learn Quickly for Few-Shot Learning. In *arXiv preprint:1707.09835*, 2017.

[23] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive Propagation Network for Few-shot Learning. In *arXiv preprint:1805.10002*, 2018.

[24] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A Simple Neural Attentive Meta-Learner. In *ICLR*, 2018.

[25] Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. In *arXiv preprint:1803.02999*, 2018.

[26] Boris N. Oreshkin, Pau Rodriguez, and Alexandre Lacoste. TADAM: Task dependent adaptive metric for improved few-shot learning. In *NIPS*, 2018.

[27] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan Yuille. Few-Shot Image Recognition by Predicting Parameters from Activations. In *CVPR*, 2018.

[28] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[29] Mengye Ren, Renjie Liao, Ethan Fetaya, and Richard S. Zemel. Incremental few-shot learning with attention attractor networks. In *arXiv preprint:1810.07218*, 2018.

[30] 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. In *ICLR*, 2018.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[33] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

[34] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. Projective subspace networks for few-shot learning. In *OpenReview*, 2019.

[35] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical Networks for Few-shot Learning. In *NIPS*, 2017.

[36] 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 *CVPR*, 2018.

[37] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[38] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In *NIPS*, 2016.

[39] Zhirong Wu, Alexei A. Efros, , and Stella X. Yu. Improving Generalization via Scalable Neighborhood Component Analysis. In *ECCV*, 2018.

[40] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.