FEW-SHOT LEARNING BY FOCUSING ON DIFFERENCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-shot classification may involve differentiating data that belongs to a different level of labels granularity. Compounded by the fact that the number of available labeled examples are scarce in the novel classification set, relying solely on the loss function to implicitly guide the classifier to separate data based on its label might not be enough; few-shot classifier needs to be very biased to perform well. In this paper, we propose a model that incorporates a simple prior: focusing on differences by building a *dissimilar* set of class representations. The model treats a class representation as a vector and removes its component that is shared among closely related class representatives. It does so through the combination of learned attention and vector orthogonalization. Our model works well on our newly introduced dataset, Hierarchical-CIFAR, that contains different level of labels granularity. It also substantially improved the performance on fine-grained classification dataset, CUB; whereas staying competitive on standard benchmarks such as mini-Imagenet, Omniglot, and few-shot dataset derived from CIFAR.

1 INTRODUCTION

Progress in artificial intelligence (AI) has been rapid. AI agents have been outperforming humans in an increasing variety of tasks, such as in recognizing images on ImageNet (He et al., 2016) and in the ancient game of Go (Silver et al., 2016). However, challenges remain – systems that outperform humans usually require learning from very large-scale data. In contrast, humans only require few examples to be able to rapidly adapt to a novel task; humans are still better *learners*. Few-shot learning methods aim to bridge this gap.

In few-shot learning, a classification algorithm is trained on multiple tasks from a domain, before being tested on a novel task. The novel task can involve classification into any subset of classes, with possibly different levels of granularity. For example, it is possible that the classification algorithms are only trained to differentiate between cats and dogs, but are tested on differentiating different breeds of dogs. The training set for the novel task is also very limited; in the extreme case, only one training example is provided for each class (called one-shot classification). A few-shot learning algorithm can learn to identify features that are important for doing well on the learned tasks – these can be transferred to the novel task as long as the domain remains similar. But with so few examples provided in the novel task and possible differences in the task granularity, few-shot learning algorithms need to be very biased to perform well. The question is then: what kind of bias (prior) is reasonable?

Our work. In this paper, we propose a model that focuses on the differences in the support set of closely related classes in assigning the class label to a new instance in the novel task. Our prior is loosely inspired by how scientists often work (Mill's method of difference): in looking for potential causes of a phenomenon, a scientist would often focus on the differences in the circumstances (features) in the instance in which the phenomenon occurred and the circumstances in instances for which the phenomenon did not occur (Mill, 1875). We use a metric learning method, constructing class representatives by averaging the feature vectors of the examples in each class. Our method focuses on the differences by removing components in each class representative that are shared with closely related classes. More specifically, it builds a set of class representatives that are orthogonal to the local average of closely related class representatives by removing its projection to the weighted average of the class representatives, where the weighting is performed using a

learned attention mechanism. Our results show that our architecturally-induced prior, coupled with the learned attention mechanism, is helpful to better handle fine-grained classification tasks, while remaining competitive on other types of datasets.

Our contributions.

1. We introduce a method that is built on one simple yet effective prior: focusing on differences, and show that it works well on classifying closely related classes. Our results show that the method achieves better performance on the fine-grained novel classification tasks constructed from the benchmark CUB dataset and on a newly created benchmark consisting of a mix of fine and coarse-grained classification tasks, Hierarchical-CIFAR. On other commonly used benchmark datasets, CIFAR-FS, *mini*-ImageNet, and Omniglot, performance is competitive with existing methods.

2. We propose a methodology to build harder few-shot learning datasets from an existing one by having different levels of labels granularity. This allows construction of more difficult datasets without requiring a very large hierarchical dataset. Through it, we build Hierarchical-CIFAR which is derived from CIFAR-100. Our empirical evaluation shows that the currently existing methods are not well equipped to handle this scenario.

2 Dissimilarity Network

2.1 Few-shot Learning

In few-shot learning, we are given a *base set* $\mathbb{B}$ and a *novel set* $\mathbb{N}$. The base set contains labeled examples from a large number of classes while novel set contains classes not found in the base set. The objective of few-shot learning is to train a classification algorithm P on the training set $\mathbb{X}_{train} = \mathbb{B}$, in such a way that it generalizes to the elements of the novel set $\mathbb{N}$. Some methods may also train on the small number of labeled examples from the novel set. In that case, the training set becomes $\mathbb{X}_{train} = \mathbb{B} \cup \mathbb{N}_{labeled}$, with labeled novel set $\mathbb{N}_{labeled} \subset \mathbb{N}$. In one-shot learning, the novel set only contains one labeled example for each class, while for k-shot learning, the novel set contains k examples for each class.

We use the *episodic training* proposed by Vinyals et al. (2016): at every step, samples some examples to form an episode $\mathbb{T} \subset \mathbb{X}_{train}$ to train the classification algorithm P. Each episode $\mathbb{T}$ consists of a small set of N-labeled examples (called support set) $\mathbb{S} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_N, y_N)\}$ and a set of M examples to be labeled (called query set) $\mathbb{Q} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_M\}$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ is a D-dimensional feature vector with label y_i (for computing the loss on the query set), simulating the conditions for learning the novel set $\mathbb{N}$.

Our method adopts the approach of metric learning (Davis et al., 2007), in which we learn a *metric space* (embedding) that works well with a classifier that classifies using the inner product in the metric space as a similarity measure. We define two levels of embedding based on how the embedding utilizes task information:

Global task embedding learns an embedding function that is optimized for all episodes that it is trained upon. The assumption is that the produced embedding will learn a meaningful and general representation that is sufficient to separate data points on the novel task without knowing what classes appear in the novel task.

Task-aware embedding removes the assumption that the learned global embedding is sufficient for the novel task. Instead, it takes the possible classes of the novel task into account. On the novel task, the embedding function will embed query set conditioned on support set which is aware of its member, giving the full context to the prediction. Our model builds an *explicit task-aware embedding* that separate a class from the weighted average of its closely-related classes. It is explicit in the sense that our model explicitly encodes such prior into its function (architecture). In contrast, *implicit task-aware embedding* only relies upon its loss function to adjust its function such that it induces separations between classes.

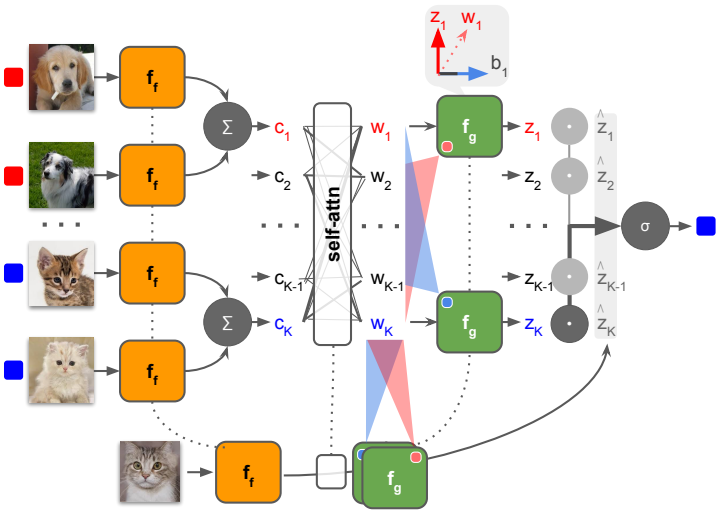

Figure 1: Dissimilarity Network architecture

2.2 MODEL

For each class, our model, which we call the Dissimilarity Network, computes a class representative, called a *prototype*, that is used to classify a new instance through the similarity (inner product) of the new instance representation with the prototype. For 1-shot classification, each training example in the novel task is transformed into the prototype, while for k-shot classification, the mean of the the k instance representations that belongs to each class transformed is into the prototype.

The Dissimilarity Network computes prototypes that are *dissimilar* to one another, in the sense that the components of the class prototypes in the direction of the weighted average of closely-related classes have been removed. This allows the model to focus on their differences in classification. The model works by iteratively enhancing the representation of the prototypes through a set of learned transformations (embedding function). The first transformation builds a global task embedding through a learned dimensionality-reduction function. The global embedding retains features that are useful for the tasks that are present in the set of training tasks (episodes) but does not take into account the classes that are present in the novel task. The second transformation transforms the class representations into a task-aware embedding using self-attention networks, taking into account other classes that are present in the task. Finally, the last transformation computes the class prototypes that are dissimilar, by locally orthogonalizing the representations to the weighted average of other closely related class representations.

When presented with a new point to classify, it computes the global embedding for that point, transform it using the task-aware embedding, locally orthogonalize the point for each possible class, and select the most similar prototype's class as the class of the new point. Figure 1 illustrates how the model construct class representations as well as predict the new point.

2.2.1 GLOBAL EMBEDDING

All our tasks are images classification tasks. For global embedding, we learn a feature extraction function $f_f : \mathbb{R}^D \rightarrow \mathbb{R}^H$ for the images. We used deep convolutional neural networks (Krizhevsky et al., 2012) which captures the local interaction of neighboring pixels and build a hierarchical representation of them. This feature extractor construct our first level of embedding, which is trained to extract information useful for all the training episodes but is not specialized to the particular novel test task.

2.2.2 SELF-ATTENTION

To give the prototypes awareness of the other class representations, we introduce the additional parameterized function $self\text{-}attn : \mathbb{R}^{K \times M} \rightarrow \mathbb{R}^{K \times M}$. By seeing the prototypes as a set of vector:

$\mathbb{C} = \{\boldsymbol{c}_1, ..., \boldsymbol{c}_K\}$, this function can learn a dynamic set-to-set operation that maximizes the objectives of the downstream operation (Section 2.2.3), which, in our case, is building a set of vectors that are locally orthogonal to its shared components. This gives *task-awareness* to our prototypes.

Our set-to-set operation is a dot-product self-attention (Vaswani et al., 2017) with embedding functions for query h_Q, key h_K, and value h_V, where each embedding function is parameterized by a neural network that computes a mapping $\mathbb{R}^{K\times M} \rightarrow \mathbb{R}^{K\times M}$. For simplicity we assume that for any input in the form of set of vector $\mathbb{A}$, it will automatically be cast into matrix $\boldsymbol{A} \in \mathbb{R}^{K\times M}$, whereas the output will be cast back into a set. The self-attention mechanism is formulated as follows.

$$self\text{-}attn(\boldsymbol{C}) = softmax\Big(\frac{h_Q(\boldsymbol{C})h_K(\boldsymbol{C})^T}{\sqrt{M}}\Big)h_V(\boldsymbol{C}) \tag{1}$$

Intuitively, the self-attention computes a weighted average of element of the input matrix $\boldsymbol{C} \in \mathbb{R}^{K\times M}$, representing set $\mathbb{K}$, which in this case is the prototypes. This operation has the effect of averaging out noisy components from global embedding that may be relevant for other tasks but are irrelevant to the set of classes in the current novel task. The weights are obtained using the learned attention function, which in this case, is parameterized by h_Q and h_K. In our case, our prototype $\boldsymbol{c}_k$ learns to be aware of the other class representations $\mathbb{C} \setminus \boldsymbol{c}_k$ by incorporating some of their components through attention.

We use bidirectional LSTM (BLSTM) (Hochreiter & Schmidhuber, 1997) for our attention embedding function for query h_Q and key h_K, while using either identity function or BLSTM on h_V. BLSM computes a concatenation of two sequence of opposing-direction by sequentially applying the element $\boldsymbol{x}_t \in \mathbb{R}^H$ of its input $\boldsymbol{x} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_T]$ into an LSTM, $\boldsymbol{h}_t, \boldsymbol{u}_t = LSTM(\boldsymbol{x}_{t-1}, \boldsymbol{h}_{t-1}, \boldsymbol{u}_{t-1})$. The computation yields a sequence of vector that each are conditioned on its neighboring elements.

2.2.3 Focusing on Differences

We encode our prior in the form of architecture inductive bias, in which we remove components that are shared with other class representations, thereby giving the model the ability to *focus* only on class differences. Since we treat the prototypes as a vector, one natural way to form such focused representation is by making the prototype locally orthogonal to a weighted average of similar components.

For an H-dimensional vector prototype $\boldsymbol{c}_k \in \mathbb{C}$, we learn to find similar components among $\mathbb{C} \setminus \boldsymbol{c}_k$ by using attention mechanism $attn : \mathbb{R}^H \times \mathbb{R}^{K\times M} \rightarrow \mathbb{R}^H$, that has previously been implicitly parameterized by the $self\text{-}attn$ (Section 2.2.2), since $\mathbb{W} = self\text{-}attn(\boldsymbol{C})$, $\boldsymbol{w} \in \mathbb{W}$, and $\boldsymbol{W}' \in \mathbb{R}^{K\times M}$ matrix representation of $\mathbb{W}' = \mathbb{W} \setminus \boldsymbol{w}$. Following the same *set-matrix assumption* from Section 2.2.2, the attention mechanism is computed as follows.

$$attn(\boldsymbol{w}, \boldsymbol{W}') = softmax\Big(\frac{\boldsymbol{w} \cdot \boldsymbol{W}'^T}{\sqrt{M}}\Big)\boldsymbol{W}' \tag{2}$$

Essentially, through weighted average, it select components among the other class prototypes based on how similar the embedded vector prototype $\boldsymbol{w}$ is to them.

We use the shared components $\boldsymbol{b}_k = attn(\boldsymbol{w}_k, \mathbb{W}' \setminus \boldsymbol{w})$ that belongs to class k as the basis of the projection $proj(\boldsymbol{w}_k, \boldsymbol{b}_k)$, which computes a mapping $\mathbb{R}^H \times \mathbb{R}^H \rightarrow \mathbb{R}^H$ as follows:

$$proj(\boldsymbol{w}, \boldsymbol{b}) = \frac{\boldsymbol{w} \cdot \boldsymbol{b}}{||\boldsymbol{b}||^2}\boldsymbol{b} \tag{3}$$

Intuitively, it *maps* the vector of prototype $\boldsymbol{w}_k$ to represent its direction using the given basis $\boldsymbol{b}_k$ (i.e., projecting it into the basis). It produces a components that are completely linearly dependent to the basis, thereby removing it: $\boldsymbol{z}_k = \boldsymbol{w}_k - proj(\boldsymbol{w}_k, \boldsymbol{b}_k)$ will give an orthogonal vector, which in this case with respect to the weighted average of the components.

2.2.4 Classification and Learning

We summarize the structure of the Dissimilarity Network here and describe how it is used for classification and learning. The network consists of two major parts: an embedding function f and

a classifier p. The embedding function $f = (f_g \circ f_f)$ is a composition of the global embedding feature extractor $f_f : \mathbb{R}^D \rightarrow \mathbb{R}^H$ and the task aware local orthogonal embedding function $f_g : \mathbb{R}^H \times \mathbb{R}^H \rightarrow \mathbb{R}^H$. It first computes the class mean (or prototype) $\boldsymbol{c}_k$ of the H-dimensional representation of the support points after global embedding,

$$\boldsymbol{c}_k = \frac{1}{|\mathbb{S}_k|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathbb{S}_k} f_f(\boldsymbol{x}_i). \tag{4}$$

For the task aware embedding, we have augmented the prototypes with task-awareness by incorporating components from other prototypes through self-attention ($self\text{-}attn$): a learned weighted averaging function. Given a set of self-attention embedded vector $\mathbb{W} = self\text{-}attn(\boldsymbol{C})$, $\boldsymbol{w}_k \in \mathbb{R}^H$: $\boldsymbol{w} \in \mathbb{W}$ that belongs to class k, we compute an H-dimensional vector of local orthogonal prototype $\boldsymbol{z}_k = f_g(\boldsymbol{w}_k, \boldsymbol{b}_k)$ using projection function $proj(\boldsymbol{w}, \boldsymbol{b})$ that projects the vector $\boldsymbol{w}$ into $\boldsymbol{b}$.

$$f_g(\boldsymbol{w}, \boldsymbol{b}) = \boldsymbol{w} - proj(\boldsymbol{w}, \boldsymbol{b}) \tag{5}$$

with $\boldsymbol{b}_k$ computed using attention mechanism $attn$ which takes the components of other prototypes $\mathbb{W} \setminus \boldsymbol{w}_k$ that are similar to $\boldsymbol{w}_k$.

$$\boldsymbol{b}_k = attn(\boldsymbol{w}_k, \mathbb{W} \setminus \boldsymbol{w}_k) \tag{6}$$

Each new prototype representation $\boldsymbol{z}_k$ is orthogonal to weighted average of the components of other closely-related prototypes.

For classification on a new point $\hat{\boldsymbol{x}}$, we first compute the global embedding for the point $\hat{\boldsymbol{v}} = f_f(\hat{\boldsymbol{x}})$ then compute the task aware embeddings $\hat{\boldsymbol{z}_k} = f_g(\hat{\boldsymbol{w}_k}, \boldsymbol{b}_k)$ for comparison with each prototype using $\hat{\mathbb{W}} = self\text{-}attn(\{\hat{\boldsymbol{v}}, ..., \hat{\boldsymbol{v}}\})$, with $\hat{\boldsymbol{w}_k} \in \mathbb{R}^H$ representing $\hat{\mathbb{W}}$ at index k, $|\hat{\mathbb{W}}| = K$. As much as possible, we embed the new point in the same way as the prototypes as it will be compared with the prototypes for classification. If the attention value embedding h_V (Section 2.2.2) is an identity function (or any other element-wise function), $\boldsymbol{w}_k = \boldsymbol{v}_k$ since its taking a weighted average of a set of identical vectors. When h_V is a BLTSM (or function that operates on a set of elements), h_V transforms the vector through multi-stage non-linear processing.

Given an unlabeled data $\hat{\boldsymbol{x}}$, the model gives a set of locally orthogonalized vector $\{\hat{\boldsymbol{z}}_1, ..., \hat{\boldsymbol{z}}_K\}$ for comparison with the prototypes. Dissimilarity Network then computes a distribution over classes for point $\hat{\boldsymbol{x}}$ using Softmax over inner product:

$$p(y = k|\hat{\boldsymbol{x}}) = \frac{\exp(\langle \hat{\boldsymbol{z}_k}, \boldsymbol{z}_k \rangle)}{\sum_{k' \neq k} \exp(\langle \hat{\boldsymbol{z}_{k'}}, \boldsymbol{z}_{k'} \rangle)} \tag{7}$$

Learning is then done using the cross-entropy loss with the label of the instance.

3 Related Works

There are many works on few-shot learning, which was started on the assumption that currently learned task can help in making a prediction in a new task (Fei-Fei et al., 2006). It soon gained interest from many researchers, which introduced many interesting techniques which contributes to huge strides of progress in few-shot learning.

Transfer learning-based methods follows the idea that the features that are learned on some datasets can be used to help classification on a novel task. It follows the standard transfer learning procedure in which involves network pre-training on the base datasets, followed by fine-tuning on the novel dataset. Gidaris & Komodakis (2018); Qi et al. (2018) propose to directly predicting weights of the classifiers on novel task. Chen et al. (2019) shows how such a simple approach is often severely underestimated, and in fact, can compete with the more complex few-shot learning technique.

Initialization based methods address few-shot learning by finding a way to better initialize a model. One line of works are concerned with how few-shot learning can be thought of learning an optimizer that can update the network rapidly and efficiently. Ravi & Larochelle (2016) uses LSTM as a meta-optimizer to rapidly adapt neural network on the novel task, whereas Munkhdalai & Yu (2017) uses external memory to update weights. Another line of works is concerned on finding a good

initialization, as such that finetuning can be done using fewer steps (Finn et al., 2017; Nichol & Schulman, 2018; Rusu et al., 2018).

Metric and similarity learning-based methods assumes that representation produced by some model on a task share some similarities with those that are produced by another task. Essentially, the goal is to learn a comparison model that can distinguish different classes on the novel task by measuring its distance (or score) or similarity to some representation produced by support set.

Our proposed method is related to prototypical networks (Snell et al., 2017) and matching network (Vinyals et al., 2016). Similar to prototypical networks (and Mensink et al. (2013)), we use mean representation of the class (prototypes) to represent each class. However, instead of relying on the assumption that each class prototype will have reasonable distance against each other, we explicitly set a simple architecture inductive bias to separate points belonging to different class apart (through orthogonalization). Moreover, instead of using euclidian distance over softmax for classification, we use the inner product scoring function which measures how related the vector representations are.

Dissimilarity Network use context embedding similar to the full context embedding (FCE) extension of the matching network, which also performed a set-to-set operation. As pointed out by Snell et al. (2017), FCE extension of the matching network does not make that much difference due to the limited number of data in few-shot learning. Here, we adopt a different architecture that supports the downstream operation to explicitly induce a local class-orthogonal representation. Instead of conditioning the prediction on the full set of support points, our task-aware embedding function only provides context for the prototypes in the form of weighted average of themselves. In the downstream operation, the embedded prototypes are orthogonalized locally, with new points being removed of their projections against local averages of prototypes that they do not belong to.

Similar to prototypical networks, our distance function (scoring function) can be set differently. One interesting extension is to use learned distance function similar to RelationNet (Sung et al., 2018).

4 EXPERIMENTS

Datasets & scenarios. We evaluate all models on the standard dataset that is widely used in few-shot learning: omniglot, miniImageNet, and CUB. Apart from that, we also evaluate all models on CIFAR dataset with the two splits: hard and normal split (which we will elaborate later).

Evaluation. Our evaluation follows Chen et al. (2019), that is by sampling N-class to form N-way classification (with N=5 unless otherwise stated). For k-shot task, we pick k labeled instances for each class to be the support set and 16 instances for query set. All results are averaged over 600 experiments which follow the above settings. We evaluate all models on 1-shot and 5-shot setting, which is the most common setting adopted in few-shot learning. To better match the inference at a test time, we use episodic training as introduced by Vinyals et al. (2016), which is also widely adopted by other (Snell et al., 2017; Chen et al., 2019; Ravi & Larochelle, 2016).

Implementation details. All methods are trained using Adam optimizer Kingma & Ba (2014) with the initial learning rate of 10^{-3}, which we cut half every 2000 episodes. We apply the following standard data augmentation on all datasets (except CIFAR): random crop, right-left flip, and color jittering. Following Snell et al. (2017), we use a four-layer convolution backbone (Conv-4) with an input size of 84x84 as a feature extractor for all methods. We use the open-source implementation of Chen et al. (2019) for other methods that we reported. We pick the best performing model based on the validation for meta-learning methods, whereas for baseline and baseline++ Chen et al. (2019), we follow the recommended settings prescribed in their paper. We trained our model on 800 epochs. Our best performing model on all datasets, based on validation set, uses identity function for the value attention embedding h_V except on the CUB dataset, which uses BLSTM. BLSTM context sharing between key attention embedding h_K and query attention embedding h_Q were found to improve performance for all datasets.

4.1 STANDARD BENCHMARKS

Omniglot (Lake et al., 2011) dataset consist of 1623 handwritten digits from 50 different alphabets. There are 20 examples per character which is drawn by different people. We follow Vinyals et al. (2016) procedure for evaluation.

***mini*-ImageNet** is a 100 classes subset of ImageNet (Deng et al., 2009) dataset (ILSVRC-12 dataset), which was first proposed by Vinyals et al. (2016). It consists of 600 images per class. Recent works follow the setting proposed by Ravi & Larochelle (2016), which consist of randomly selected 64 base, 16 validation, and 20 novel classes.

CUB or Caltech-UCSD Birds 200-2011 dataset (Wah et al., 2011) is a fine-grained classification dataset which consist of 200 classes (bird species) and 11,788 images in total. We follow Hilliard et al. (2018) setting which is composed of 100 base, 50 validation, and 50 novel classes.

CIFAR-FS dataset is derived from CIFAR-100 dataset (Krizhevsky et al., 2009) consist of 60,000 32x32 color images with 100 classes (belonging to 20 superclasses), with 600 images for each class. Our split consists of randomly sampled 40 base, 15 validation, and 45 novel classes (detail of the split can be found on appendix).

Results. Table 1 shows how our method fares against others. Our method performs the best on a fine-grained classification task such as CUB and improved on a wide margin on its 1-shot classification task. As we have suspected, there is an increasing need to focus on the difference between classes when the classification task becomes increasingly fine-grained. Despite also being trained on fine-grained classification task on CUB dataset, our prior still seems to help classify similar-looking yet different classes as it further separate class representation by explicitly removing latent features shared among classes. On CIFAR-FS, the Dissimilarity Network are slightly better than the rest, whereas being competitive on Omniglot and *mini*-ImageNet dataset. Overall, our method surpasses the state-of-the-art methods on average, on both 1-shot and 5-shot classification task.

4.2 HARDER BENCHMARKS

General setting. Assume a J-labeled dataset $\mathbb{D} = \{(\boldsymbol{x}_1, y_1^{coarse}, y_1^{fine}), ..., (\boldsymbol{x}_J, y_J^{coarse}, y_J^{fine})\}$ where the labels comes from a two-level hierarchy: coarse-grained label $y_i^{coarse} \in \mathbb{K}^{coarse}$ and fine-grained label $y_i^{fine} \in \mathbb{K}^{fine}$. $\mathbb{K}_s^{fine}$ denote a subset of labels $\mathbb{K}^{fine}$ that belongs to coarse-grained label (superclass) $s \in \mathbb{K}^{coarse}$. For example, Siamese, Persian, and Ragdol belongs to superclass Cat, and we refer to them as subclasses of Cat. The base set $\mathbb{B}$ and novel set $\mathbb{N}$ can be derived using the following method (illustrated in Figure 2).

Method. For all coarse-grained label $y_i^{coarse} \in \mathbb{K}^{coarse}$, select some $y_i^{fine} \in \mathbb{K}^{fine}$ that is the subclasses of y_i^{coarse} (i.e., $\mathbb{K}_{y_i^{coarse}}^{fine}$), producing a set of fine-grained labels from all superclasses $\mathbb{K}_{base}^{fine}$. Construct the base set: $\mathbb{B} = \{(\boldsymbol{x}_i, y_i^{coarse}) | (\boldsymbol{x}_i, y_i^{coarse}, y_i^{fine}) \in \mathbb{D}, y_i^{fine} \in \mathbb{K}_{base}^{fine}\}$. The novel set can be built by taking the rest of unused data and pair them with their fine-grained labels: $\mathbb{N} = \{(\boldsymbol{x}_i, y_i^{fine}) | (\boldsymbol{x}_i, y_i^{coarse}, y_i^{fine}) \in \mathbb{D}, y_i^{fine} \notin \mathbb{K}_{base}^{fine}\}$. Validation set is constructed the same way as novel set. We leave out the detail of its construction for simplicity.

It's trivial to see that this approach can work on smaller datasets. This approach is advantageous as on each task, the labels can vary from being fine-grained to coarse-grained depending on the random selection. As such, the methods that rely on the awareness of the overall novel tasks will likely to fail as it builds a general embedding that works on all but not optimized for the current task. On the other hand, a dynamic method that conditions the prediction on the support set of the given task will likely to perform better.

Hierarchical-CIFAR is derived from CIFAR-100, through the use of the aformentioned method. The following is the detail for each split (more detail on the appendix). 20 coarse-grained base classes from 40 fine-grained classes (derived from the entire 20 superclasses, 2 classes each). 15 validation classes (derived from 5 superclass, 3 classes each). 45 novel classes (derived from 15 superclass, 5 classes each).

Results. From Table 1, we can compare the performance of all models on the randomly-split CIFAR-FS dataset and hierarchically split Hierarchical-CIFAR dataset. All methods perform substantially worse on the latter dataset, confirming the dataset to be, in fact, harder.

As can be seen from the results, our method substantially outperforms others on the more difficult CUB and Hierarchical-CIFAR datasets while giving comparable performance on the other datasets.

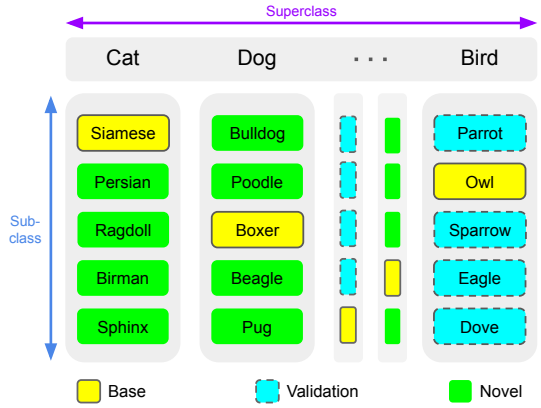

Figure 2: Illustration on construction of harder dataset. The base dataset uses the superclass taxonomy (e.g., cat, dog, and bird) instead its subclass (e.g., siamese, boxer, and owl).

Table 1: 5-shot and 1-shot classification accuracy on all datasets. We test our method against MatchingNet (Vinyals et al., 2016), ProtoNet (Snell et al., 2017), MAML (Finn et al., 2017), RelationNet (Sung et al., 2018), and Baseline++ (Chen et al., 2019). Our method consistently surpassed state-of-the-art methods on Hierarchical-CIFAR (H-CIFAR), CUB, and CIFAR-FS dataset, whereas being competitive on *mini*-ImageNet and Omniglot dataset. All accuracy results are averaged over 600 test episodes and are reported with 95% confidence intervals. *Results reported by Chen et al. (2019).

	5-shot classification accuracy (%)					
Method	**H-CIFAR**	**CUB**	**CIFAR-FS**	***mini*-ImageNet**	**Omniglot**	**Average**
DissimilarityNet (Ours)	68.45 ± 0.74	81.23 ± 0.63	71.10 ± 0.71	65.40 ± 0.61	99.27 ± 0.10	77.09 ± 0.56
MatchingNet	63.34 ± 0.78	72.86 ± 0.70*	67.14 ± 0.77	63.48 ± 0.66*	99.37 ± 0.11*	73.24 ± 0.60
ProtoNet	64.30 ± 0.81	70.77 ± 0.69*	69.96 ± 0.77	64.24 ± 0.72*	99.15 ± 0.12*	73.68 ± 0.62
MAML	62.87 ± 0.77	72.09 ± 0.76*	65.98 ± 0.81	62.71 ± 0.71*	99.53 ± 0.08*	72.64 ± 0.63
RelationNet	63.15 ± 0.83	76.11 ± 0.69*	68.87 ± 0.76	66.60 ± 0.69*	99.30 ± 0.10*	74.81 ± 0.61
Baseline++	57.25 ± 0.77	79.34 ± 0.61*	59.86 ± 0.80	66.43 ± 0.63*	99.38 ± 0.10*	72.45 ± 0.58
	1-shot classification accuracy (%)					
Method	**H-CIFAR**	**CUB**	**CIFAR-FS**	***mini*-ImageNet**	**Omniglot**	**Average**
DissimilarityNet (Ours)	51.02 ± 0.89	65.82 ± 0.94	54.66 ± 0.82	49.34 ± 0.78	97.90 ± 0.25	63.75 ± 0.74
MatchingNet	50.42 ± 0.92	61.16 ± 0.89*	53.92 ± 0.92	48.14 ± 0.78*	97.78 ± 0.30*	62.28 ± 0.76
ProtoNet	47.16 ± 0.90	51.31 ± 0.91*	50.08 ± 0.88	44.42 ± 0.84*	98.01 ± 0.30*	58.20 ± 0.77
MAML	48.64 ± 0.93	55.92 ± 0.95*	51.78 ± 0.94	46.47 ± 0.82*	98.57 ± 0.19*	60.28 ± 0.77
RelationNet	50.78 ± 0.95	62.45 ± 0.98*	54.24 ± 0.93	49.31 ± 0.85*	97.22 ± 0.33*	62.80 ± 0.81
Baseline++	39.30 ± 0.70	60.53 ± 0.83*	43.38 ± 0.73	48.24 ± 0.75*	95.41 ± 0.39*	57.37 ± 0.68

5 Conclusion

We have proposed Dissimilarity Network for few-shot learning based on the idea of focusing on differences in the class representation. Our approach directly addresses the failure modes of some few-shot classifiers that do not explicitly take into account the classification task at hand, yielding non-satisfactory results on some task such as fine-grained novel classification with coarse-grained base classification task. To demonstrate the necessities of building task-aware embedding for such task, we came up with a challenging dataset, Hierarchical CIFAR, which we have shown to be harder than the CIFAR-FS. Dissimilarity Network introduced an architecture inductive bias which removes the shared components among classes in the prototypes by orthogonalizing them (i.e., removing their projected components) to their leave-self-out weighted local average. Our method performs comparably to the state-of-the-art methods on standard benchmarks such as Omniglot and *mini*-ImageNet, and substantially outperform other methods on CUB dataset and on the newly constructed Hierarchical CIFAR dataset.

REFERENCES

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pp. 209–216. ACM, 2007.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.

John Stuart Mill. A system of logic, ratiocinative and inductive: Being a connected view of the principles of evidence, and the methods of scientific investigation, vol. 1. 1875.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2554–2563. JMLR. org, 2017.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018.

Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5822–5830, 2018.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

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*, pp. 5998–6008, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

A Appendix

A.1 Splits for CIFAR

Base: forest, house, television, wolf, cloud, sweet_pepper, dinosaur, tank, caterpillar, cup, sunflower, whale, can, bottle, road, crocodile, woman, bear, otter, willow_tree, snail, aquarium_fish, girl, trout, bowl, worm, pear, streetcar, castle, flatfish, lobster, turtle, poppy, orchid, man, seal, lamp, lawn_mower, beetle, clock

Validation: oak_tree, kangaroo, mushroom, porcupine, squirrel, lizard, train, spider, keyboard, maple_tree, bicycle, orange, lion, rabbit, motorcycle

Novel: fox, boy, skyscraper, bridge, mouse, shrew, plain, possum, tiger, tulip, wardrobe, sea, couch, mountain, leopard, camel, shark, plate, dolphin, table, bee, pickup_truck, palm_tree, beaver, baby, bus, butterfly, ray, apple, cattle, crab, pine_tree, raccoon, tractor, chair, rose, telephone, chimpanzee, snake, bed, hamster, skunk, cockroach, rocket, elephant

A.2 Splits for Hierarchical CIFAR

Base: aquatic_mammals, fish, flowers, food_containers, fruit_and_vegetables, household_electrical_devices, household_furniture, insects, large_carnivores, large_man-made_outdoor_things, large_natural_outdoor_scenes, large_omnivores_and_herbivores, medium_mammals, non-insect_invertebrates, people, reptiles, small_mammals, trees, vehicles_1, vehicles_2

Validation: rabbit, hamster, bed, house, kangaroo, lamp, skyscraper, squirrel, castle, table, chimpanzee, telephone, television, wardrobe, elephant

Novel: baby, beaver, beetle, bicycle, bottle, bus, butterfly, can, caterpillar, crocodile, cup, dolphin, flatfish, forest, girl, lion, lobster, man, mountain, oak_tree, orange, orchid, otter, pear, pickup_truck, pine_tree, porcupine, possum, ray, rocket, rose, sea, shark, skunk, snail, snake, streetcar, sweet_pepper, tank, tiger, tulip, turtle, willow_tree, wolf, worm