# Few-Shot Learning via Embedding Adaptation with Set-to-Set Functions

Han-Jia Ye*
Nanjing University
yehj@lamda.nju.edu.cn

Hexiang Hu
USC
hexiangh@usc.edu

De-Chuan Zhan
Nanjing University
zhandc@lamda.nju.edu.cn

Fei Sha†
USC & Google
fsha@google.com

## Abstract

*Learning with limited data is a key challenge for visual recognition. Many few-shot learning methods address this challenge by learning an instance embedding function from seen classes and apply the function to instances from unseen classes with limited labels. This style of transfer learning is task-agnostic: the embedding function is not learned optimally discriminative with respect to the unseen classes, where discerning among them leads to the target task. In this paper, we propose a novel approach to adapt the instance embeddings to the target classification task with a* set-to-set *function, yielding embeddings that are task-specific and are discriminative. We empirically investigated various instantiations of such set-to-set functions and observed the Transformer is most effective — as it naturally satisfies key properties of our desired model. We denote this model as* FEAT *(few-shot embedding adaptation w/ Transformer) and validate it on both the standard few-shot classification benchmark and four extended few-shot learning settings with essential use cases,* i.e.*, cross-domain, transductive, generalized few-shot learning, and low-shot learning. It archived consistent improvements over baseline models as well as previous methods, and established the new state-of-the-art results on two benchmarks.*

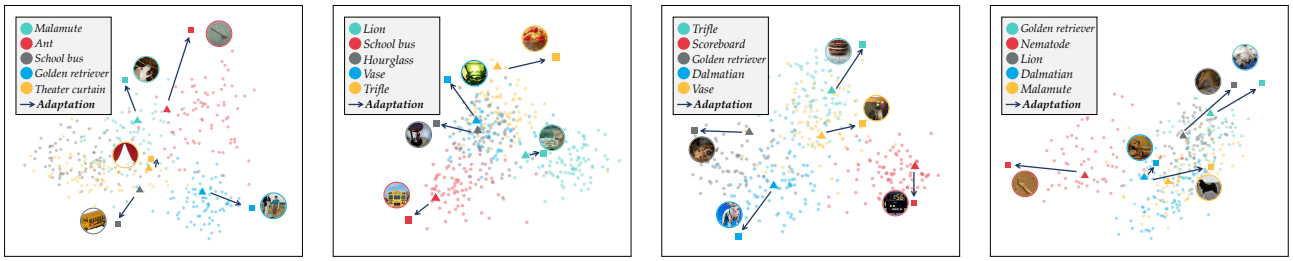
## 1. Introduction

Few-shot visual recognition [9, 18, 19, 22, 43] emerged as a promising direction in tackling the challenge of learning new visual concepts with limited annotations. Concretely, it distinguishes two sets of visual concepts: SEEN and UNSEEN ones. The target task is to construct visual classifiers to identify classes from the UNSEEN where each class has a very small number of exemplars ("few-shot"). The main idea is to discover transferable visual knowledge in the SEEN classes, which have ample labeled instances, and leverage it to construct the desired classifier. For example, state-of-the-art approaches for few-shot learning [35, 38, 40, 43] usually learn a discriminative instance embedding model on the SEEN categories, and apply it to visual data in UNSEEN categories. In this common embedding space, non-parametric classifiers (*e.g.*, nearest neighbors) are then used to avoid learning complicated recognition models from a small number of examples.

Such approaches suffer from one important limitation. Assuming a common embedding space implies that the discovered knowledge – discriminative visual features – on the SEEN classes are equally effective for *any* classification tasks constructed for an arbitrary set of UNSEEN classes. In concrete words, suppose we have two different target tasks: discerning "cat" versus "dog" and discerning "cat" versus "tiger". Intuitively, each task uses a different set of discriminative features. Thus, the most desired embedding model first needs to be able to extract discerning features for either task at the same time. This could be a challenging aspect in its own right as the current approaches are agnostic to what those "downstream" target tasks are and could accidentally de-emphasize selecting features for future use. Secondly, even if both sets of discriminative features are extracted, they do not necessarily lead to the optimal performance for a *specific* target task. The most useful features for discerning "cat" versus "tiger" could be irrelevant and noise to the task of discerning "cat" versus "dog"!

What is missing from the current few-shot learning approaches is an *adaptation* strategy that tailors the visual knowledge extracted from the SEEN classes to the UNSEEN ones in a target task. In other words, we desire separate embedding spaces where each one of them is customized such that the visual features are most discriminative for a given task. Towards this, we propose a few-shot model-based embedding adaptation method that adjusts the instance embedding models derived from the SEEN classes. Such model-based embedding adaptation requires a *set-to-set function*: a function mapping that takes all instances from the few-shot support set and outputs the set of adapted support instance embeddings, with elements in the set co-adapting with each other. Such output embeddings are then assembled as the prototypes for each visual category and serve as the nearest neighbor classifiers. Figure 1 qualitatively illustrates

*Work mostly done when the author was a visiting scholar at USC.
†On leave from USC




**(a)** Acc↑: 40.33% → 55.33% **(b)** Acc↑: 48.00% → 69.60% **(c)** Acc↑: 43.60% → 63.33% **(d)** Acc↓: 56.33% → 47.13%

**Figure 1: Qualitative visualization** of model-based embedding adaptation procedure (implemented using FEAT) on test tasks (refer to § 5.2.2 for more details). Each figure shows the locations of PCA projected support embeddings (class prototypes) before and after the adaptation of FEAT. Values below are the 1-shot 5-way classification accuracy before and after the the adaptation. Interestingly, the embedding adaptation step of FEAT pushes the support embeddings apart from the clutter and toward their own clusters, such that they can better fits the test data of its categories. (Best view in colors!)

the embedding adaptation procedure (as results of our best model). These class prototypes spread out in the embedding space toward the samples cluster of each category, indicating the effectiveness of embedding adaptation.

In this paper, we implement the set-to-set transformation using a variety of function approximators, including bidirectional LSTM [12] (Bi-LSTM), deep sets [49], graph convolutional network (GCN) [16], and Transformer [24, 41]. Our experimental results (refer to § 5.2.1) suggest that Transformer is the most parameter efficient choice that at the same time best implements the key properties of the desired set-to-set transformation, including *contextualization*, *permutation invariance*, *interpolation* and *extrapolation* capabilities (see § 4.1). As a consequence, we choose the set-to-set function instantiated with Transformer to be our final model and denote it as FEAT (**F**ew-shot **E**mbedding **A**daptation with **T**ransformer). We further conduct comprehensive analysis on FEAT and evaluate it on many extended tasks, including few-shot domain generalization, transductive few-shot learning, and generalized few-shot learning.

Our overall contribution is three-fold.

- We formulate the few-shot learning as a model-based embedding adaptation to make instance embeddings task-specific, via using a set-to-set transformation.
- We instantiate such set-to-set transformation with various function approximators, validating and analyzing their few-shot learning ability, task interpolation ability, and extrapolation ability, *etc*. It concludes our model (FEAT) that uses the Transformer as the set-to-set function.
- We evaluate our FEAT model on a variety of extended few-shot learning tasks, where it achieves superior performances compared with strong baseline approaches.

## 2. Related Work

Methods specifically designed for few-shot learning fall broadly into two categories. The first is to control how a classifier for the target task should be constructed. One fruitful idea is the meta-learning framework where the classifiers are optimized *in anticipation* that a future update due to data from a new task performs well on that task [2, 3, 9, 10, 21, 27, 31, 35], or the classifier itself is directly meta-predicted by the new task data [30, 47].

Another line of approach has focused on learning generalizable instance embeddings [1, 4, 5, 13, 17, 26, 37, 40, 43] and uses those embeddings on simple classifiers such as nearest neighbor rules. The key assumption is that the embeddings capture all necessarily discriminative representations of data such that simple classifiers are sufficed, hence avoiding the danger of overfitting on a small number of labeled instances. Early work such as [17] first validated the importance of embedding in one-shot learning, whilst [43] proposes to learn the embedding with a soft nearest neighbor objective, following a meta-learning routine. Recent advances have leveraged different objective functions for learning such embedding models, *e.g*., considering the class prototypes [38], decision ranking [40], and similarity comparison [39]. Most recently, [36] utilizes the graph convolution network [16] to unify the embedding learning.

Our work follows the second school of thoughts. The main difference is that we do not assume the embeddings learned on SEEN classes, being agnostic to the target tasks, are necessarily discriminative for those tasks. In contrast, we propose to *adapt* those embeddings for each target task *with a set-to-set function* so that the transformed embeddings are better aligned with the discrimination needed in those tasks. We show empirically that such task-specific embeddings perform better than task-agnostic ones. MetaOptNet [20] and CTM [23] follow the same spirit of learning task-specific embedding (or classifiers) via either explicitly optimization of target task or using concentrator and projector to make distance metric task-specific.

## 3. Learning Embedding for Task-agnostic FSL

In the standard formulation of few-shot learning (FSL) [9, 43], a task is represented as a $M$-shot $N$-way

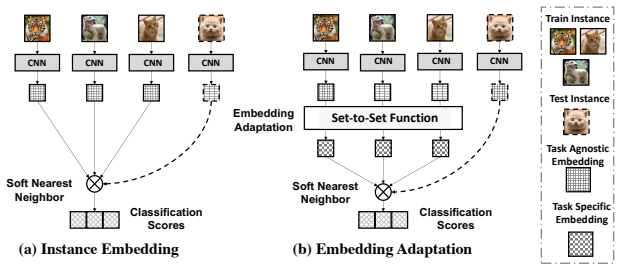



**Figure 2:** Illustration of the proposed **F**ew-Shot **E**mbedding **A**daptation **T**ransformer (FEAT). Existing methods usually use the same embedding function $\mathbf{E}$ for all tasks. We propose to adapt the embeddings to each target few-shot learning task with a set-to-set function such as Transformer, BiLSTM, DeepSets, and GCN.

classification problem with $N$ classes sampled from a set of visual concepts $\mathcal{U}$ and $M$ (training/support) examples per class. We denote the training set (also referred as support sets in the literature) as $\mathcal{D}_{\mathbf{train}} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{NM}$, with the instance $\mathbf{x}_i \in \mathbb{R}^D$ and the one-hot labeling vector $\mathbf{y}_i \in \{0,1\}^N$. We will use "support set" and "training set" interchangeably in the paper. In FSL, $M$ is often small (*e.g.*, $M = 1$ or $M = 5$). The goal is to find a function $f$ that classifies a test instance $\mathbf{x}_{\mathbf{test}}$ by $\hat{\mathbf{y}}_{\mathbf{test}} = f(\mathbf{x}_{\mathbf{test}}; \mathcal{D}_{\mathbf{train}}) \in \{0,1\}^N$.

Given a small number of training instances, it is challenging to construct complex classifiers $f(\cdot)$. To this end, the learning algorithm is also supplied with additional data consisting of ample labeled instances. These additional data are drawn from visual classes $\mathcal{S}$, which does not overlap with $\mathcal{U}$. We refer to the original task as *the target task* which discerns $N$ UNSEEN classes $\mathcal{U}$. To avoid confusion, we denote the data from the SEEN classes $\mathcal{S}$ as $\mathcal{D}^{\mathcal{S}}$.

To learn $f(\cdot)$ using $\mathcal{D}^{\mathcal{S}}$, we synthesize many $M$-shot $N$-way FSL tasks by sampling the data in the meta-learning manner [9, 43]. Each sampling gives rise to a task to classify a test set instance $\mathbf{x}^{\mathcal{S}}_{\mathbf{test}}$ into one of the $N$ SEEN classes by $f(\cdot)$, where the test instances set $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$ is composed of the labeled instances with the same distribution as $\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}$. Formally, the function $f(\cdot)$ is learnt to minimize the averaged error over those sampled tasks

$$f^* = \arg\min_f \sum_{(\mathbf{x}^{\mathcal{S}}_{\mathbf{test}}, \mathbf{y}^{\mathcal{S}}_{\mathbf{test}}) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \ell(f(\mathbf{x}^{\mathcal{S}}_{\mathbf{test}}; \mathcal{D}^{\mathcal{S}}_{\mathbf{train}}), \mathbf{y}^{\mathcal{S}}_{\mathbf{test}}) \tag{1}$$

where the loss $\ell(\cdot)$ measures the discrepancy between the prediction and the true label. For simplicity, we have assumed we only synthesize one task with test set $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$. The optimal $f^*$ is then applied to the original target task.

We consider the approach based on learning embeddings for FSL [38, 43] (see Figure 2 (a) for an overview). In particular, the classifier $f(\cdot)$ is composed of two elements. The first is an embedding function $\phi_{\mathbf{x}} = \mathbf{E}(\mathbf{x}) \in \mathbb{R}^d$ that maps an instance $\mathbf{x}$ to a representation space. The second component applies the nearest neighbor classifiers in this space:

$$\begin{aligned}\hat{\mathbf{y}}_{\mathbf{test}} &= f(\phi_{\mathbf{x}_{\mathbf{test}}}; \{\phi_{\mathbf{x}}, \forall(\mathbf{x},\mathbf{y}) \in \mathcal{D}_{\mathbf{train}}\}) \\ &\propto \exp\left(\mathbf{sim}(\phi_{\mathbf{x}_{\mathbf{test}}}, \phi_{\mathbf{x}})\right) \cdot \mathbf{y}, \forall(\mathbf{x},\mathbf{y}) \in \mathcal{D}_{\mathbf{train}}\end{aligned} \tag{2}$$

Note that only the embedding function is learned by optimizing the loss in Eq. 1. For reasons to be made clear in below, we refer this embedding function as *task-agnostic*.

**Algorithm 1** Training strategy of embedding adaptation

**Require:** Seen class set $\mathcal{S}$

**for all** iteration = 1,...,MaxIteration **do**
  Sample $N$-way $M$-shot $(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \mathcal{D}^{\mathcal{S}}_{\mathbf{test}})$ from $\mathcal{S}$
  Compute $\phi_{\mathbf{x}} = \mathbf{E}(\mathbf{x})$, for $\mathbf{x} \in \mathcal{X}^{\mathcal{S}}_{\mathbf{train}} \cup \mathcal{X}^{\mathcal{S}}_{\mathbf{test}}$
  **for all** $(\mathbf{x}^{\mathcal{S}}_{\mathbf{test}}, \mathbf{y}^{\mathcal{S}}_{\mathbf{test}}) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$ **do**
    Compute $\{\psi_{\mathbf{x}}\ ; \forall \mathbf{x} \in \mathcal{X}^{\mathcal{S}}_{\mathbf{train}}\}$ with $\mathbf{T}$ via Eq. 3
    Predict $\hat{\mathbf{y}}^{\mathcal{S}}_{\mathbf{test}}$ with $\{\psi_{\mathbf{x}}\}$ as Eq. 4
    Compute $\ell(\hat{\mathbf{y}}^{\mathcal{S}}_{\mathbf{test}}, \mathbf{y}^{\mathcal{S}}_{\mathbf{test}})$ with Eq. 1
  **end for**
  Compute $\nabla_{\mathbf{E},\mathbf{T}} \sum_{(\mathbf{x}^{\mathcal{S}}_{\mathbf{test}}, \mathbf{y}^{\mathcal{S}}_{\mathbf{test}}) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \ell(\hat{\mathbf{y}}^{\mathcal{S}}_{\mathbf{test}}, \mathbf{y}^{\mathcal{S}}_{\mathbf{test}})$
  Update $\mathbf{E}$ and $\mathbf{T}$ with $\nabla_{\mathbf{E},\mathbf{T}}$ use SGD
**end for**
**return** Embedding function $\mathbf{E}$ and set function $\mathbf{T}$.

## 4. Adapting Embedding for Task-specific FSL

In what follows, we describe our approach for few-shot learning (FSL). We start by describing the main idea (§ 4.1, also illustrated in Figure 2), then introduce the set-to-set adaptation function (§ 4.2). Last are learning (§ 4.3) and implementations details (§ 4.4).

### 4.1. Adapting to Task-Specific Embeddings

The key difference between our approach and traditional ones is to learn *task-specific* embeddings. We argue that the embedding $\phi_{\mathbf{x}}$ is not ideal. In particular, the embeddings do not necessarily highlight the most discriminative representation for a specific target task. To this end, we introduce an adaption step where the embedding function $\phi_{\mathbf{x}}$ (more precisely, its values on instances) is transformed. This transformation is a *set-to-set* function that *contextualizes* over the image instances of a set, to enable strong co-adaptation of each item. Instance functions fails to have such co-adaptation property. Furthermore, the set-to-set-function receives instances as bags, or sets without orders, requiring the function to output the set of refined instance embeddings while being *permutation-invariant*. Concretely,

$$\begin{aligned}\{\psi_{\mathbf{x}}\ ; \forall \mathbf{x} \in \mathcal{X}_{\mathbf{train}}\} &= \mathbf{T}\left(\{\phi_{\mathbf{x}}\ ; \forall \mathbf{x} \in \mathcal{X}_{\mathbf{train}}\}\right) \\ &= \mathbf{T}\left(\pi\left\{\phi_{\mathbf{x}}\ ; \forall \mathbf{x} \in \mathcal{X}_{\mathbf{train}}\right\}\right)\end{aligned} \tag{3}$$

where $\mathcal{X}_{\mathbf{train}}$ is a set of all the instances in the training set $\mathcal{D}_{\mathbf{train}}$ for the target task. $\pi(\cdot)$ is a permutation operator over a set. Thus the set of *adapted* embedding will not change if we apply a permutation over the input embedding set. With *adapted* embedding $\psi_{\mathbf{x}}$, the test instance $\mathbf{x}_{\mathbf{test}}$ can be classified by computing nearest neighbors w.r.t. $\mathcal{D}_{\mathbf{train}}$:

$$\hat{\mathbf{y}}_{\mathbf{test}} = f(\phi_{\mathbf{x}_{\mathbf{test}}}; \{\psi_{\mathbf{x}}, \forall(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\mathbf{train}}\}) \quad (4)$$

Our approach is generally applicable to different types of task-agnostic embedding function $\mathbf{E}$ and similarity measure $\mathbf{sim}(\cdot, \cdot)$, *e.g.*, the (normalized) cosine similarity [43] or the negative distance [38]. Both the embedding function $\mathbf{E}$ and the set transformation function $\mathbf{T}$ are optimized over synthesized FSL tasks sampled from $\mathcal{D}^{\mathcal{S}}$, sketched in Alg. 1. Its key difference from conventional FSL is in the *line 4 to line 8* where the embeddings are transformed.

### 4.2. Embedding Adaptation via Set-to-set Functions

Next, we explain various choices as the instantiations of the set-to-set embedding adaptation function.

**Bidirectional LSTM (BILSTM) [12, 43]** is one of the common choice to instantiate the set-to-set transformation, where the addition between the input and the hidden layer outputs of each BILSTM cell leads to the adapted embedding. It is notable that the output of the BILSTM suppose to depend on the order of the input set. Note that using BILSTM as embedding adaptation model is similar but different from the fully conditional embedding [43], where the later one contextualizes both training and test instance embedding altogether, which results in a transductive setting.

**DeepSets [49]** is inherently a permutation-invariant transformation function. It is worth noting that DEEPSETS aggregates the instances in a set into a holistic *set vector*. We consider two components to implement such DeepSets transformation, an instance centric *set vector* combined with a set context vector. For $\mathbf{x} \in \mathcal{X}_{\mathbf{train}}$, we define its complementary set as $\mathbf{x}^{\complement}$. Then we implement the DEEPSETS by:

$$\psi_{\mathbf{x}} = \phi_{\mathbf{x}} + g([\phi_{\mathbf{x}}; \sum_{\mathbf{x}_{i'} \in \mathbf{x}^{\complement}} h(\phi_{\mathbf{x}_{i'}})]) \quad (5)$$

In Eq. 5, $g$ and $h$ are two-layer multi-layer perception (MLP) with ReLU activation which map the embedding into another space and increase the representation ability of the embedding. For each instance, embeddings in its complementary set is first combined into a *set vector* as the context, and then this vector is concatenated with the input embedding to obtain the residual component of adapted embedding. This conditioned embedding takes other instances in the set into consideration, and keeps the "set (permutation invariant)" property. In practice, we find using the maximum operator in Eq. 5 works better than the sum operator suggested in [49].

**Graph Convolutional Networks (GCN) [16, 36]** propagate the relationship between instances in the set. We first construct the degree matrix $A$ to represent the similarity between instances in a set. If two instances come from the same class, then we set the corresponding element in $A$ to 1, otherwise to 0. Based on $A$, we build the "normalized" adjacency matrix $S$ for a given set with added self-loops $S = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$. $I$ is the identity matrix, and $D$ is the diagonal matrix whose elements are equal to the sum of elements in the corresponding row of $A + I$.

Let $\Phi^0 = \{\phi_{\mathbf{x}} \; ; \; \forall \mathbf{x} \in \mathcal{X}_{\mathbf{train}}\}$, the relationship between instances could be propagated based on $S$, *i.e.*,

$$\Phi^{t+1} = \mathbf{ReLU}(S\Phi^t W) \; , \; t = 0, 1, \ldots, T-1 \quad (6)$$

$W$ is a projection matrix for feature transformation. In GCN, the embedding in the set is transformed based on Eq. 6 multiple times, and the final $\Phi^T$ gives rise to the $\{\psi_{\mathbf{x}}\}$.

**Transformer. [41]** We use the *Transformer* architecture [41] to implement $\mathbf{T}$. In particular, we employ self-attention mechanism [24, 41] to transform each instance embedding with consideration to its contextual instances. Note that it naturally satisfies the desired properties of $\mathbf{T}$ because it outputs refined instance embeddings and is permutation invariant. We denote it as **F**ew-Shot **E**mbedding **A**daptation with **T**ransformer (FEAT).

Transformer is a store of triplets in the form of (query $\mathcal{Q}$, key $\mathcal{K}$, and value $\mathcal{V}$). To compute proximity and return values, those points are first linearly mapped into some space $K = W_K^\top \left[ \; \phi_{\mathbf{x}_k}; \forall \mathbf{x}_k \in \mathcal{K} \; \right] \in \mathbb{R}^{d \times |\mathcal{K}|}$, which is also the same for $\mathcal{Q}$ and $\mathcal{V}$ with $W_Q$ and $W_V$ respectively. Transformer computes what is the right value for a query point — the query $\mathbf{x}_q \in \mathcal{Q}$ is first matched against a list of keys $K$ where each key has a value $V$. The final value is then returned as the sum of all the values *weighted* by the proximity of the key to the query point, *i.e.* $\psi_{\mathbf{x}_q} = \phi_{\mathbf{x}_q} + \sum_k \alpha_{qk} V_{:,k}$, where

$$\alpha_{qk} \propto \exp\left(\frac{\phi_{\mathbf{x}_q}^\top W_Q \cdot K}{\sqrt{d}}\right)$$

and $V_{:,k}$ is the $k$-th column of $V$. In the standard FSL setup, we have $\mathcal{Q} = \mathcal{K} = \mathcal{V} = \mathcal{X}_{\mathbf{train}}$.

### 4.3. Contrastive Learning of Set-to-Set Functions

To facilitate the learning of embedding adaptation, we apply a contrastive objective in addition to the general one. It is designed to make sure that instances embeddings *after adaptation* is similar to the same class neighbors and dissimilar to those from different classes. Specifically, the embedding adaptation function $\mathbf{T}$ is applied to instances of each $n$ of the $N$ class in $\mathcal{D}^{\mathcal{S}}_{\mathbf{train}} \cup \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$, which gives rise to the transformed embedding $\psi'_{\mathbf{x}}$ and class centers $\{\mathbf{c}_n\}_{n=1}^N$.

Then we apply the contrastive objective to make sure training instances are close to its own class center than other centers. The total objective function (together with Eq. 1) is shown as following:

$$\mathcal{L}(\hat{\mathbf{y}}_{\mathbf{test}}, \mathbf{y}_{\mathbf{test}}) = \ell(\hat{\mathbf{y}}_{\mathbf{test}}, \mathbf{y}_{\mathbf{test}}) + \lambda \cdot \ell\big(\mathbf{softmax}\left(\mathbf{sim}(\psi'_{\mathbf{x}_{\mathbf{test}}}, \mathbf{c}_n)\right), \mathbf{y}_{\mathbf{test}}\big) \quad (7)$$

This contrastive learning makes the set transformation extract common characteristic for instances of the same category, so as to preserve the category-wise similarity.

### 4.4. Implementation details

We consider three different types of convolutional networks as the backbone for instance embedding function $\mathbf{E}$: 1) A 4-layer convolution network (ConvNet) [38, 40, 43] and 2) the 12-layer residual network (ResNet) used in [20], and 3) the Wide Residual Network (WideResNet) [35, 48]. We apply an additional pre-training stage for the backbones over the SEEN classes, based on which our re-implemented methods are further optimized. To achieve more precise embedding, we average the same-class instances in the training set before the embedding adaptation with the set-to-set transformation. Adam [15] and SGD are used to optimize ConvNet and ResNet variants respectively. Moreover, we follow the most standard implementations for the four set-to-set functions — BiLSTM [12], DeepSets [49], Graph Convolutional Networks (GCN) [16] and Transformer (FEAT) [41]. We refer readers to supplementary material (SM) for complete details and ablation studies of each set-to-set functions. Our implementation is available at https://github.com/Sha-Lab/FEAT.

## 5. Experiments

In this section, we first evaluate a variety of models for embedding adaptation in § 5.2 with standard FSL. It concludes that FEAT (with Transformer) is the most effective approach among different instantiations. Next, we perform ablation studies in § 5.2.2 to analyze FEAT in details. Eventually, we evaluate FEAT on many extended few-shot learning tasks to study its general applicability (§ 5.3). This study includes few-shot domain generalization, transductive few-shot learning, generalized few-shot learning, and large-scale low-shot learning (refer to SM).

### 5.1. Experimental Setups

**Datasets.** *Mini*ImageNet [43] and *Tiered*ImageNet [33] datasets are subsets of the ImageNet [34]. *Mini*ImageNet includes a total number of 100 classes and 600 examples per class. We follow the setup provided by [31], and use 64 classes as SEEN categories, 16 and 20 as two sets of UNSEEN categories for model validation and evaluation respectively. *Tiered*ImageNet is a large-scale dataset with more categories, which contains 351, 97, and 160 categories for model training, validation, and evaluation, respectively. In addition to these, we investigate the OfficeHome [42] dataset to validate the generalization ability of FEAT across domains. There are four domains in OfficeHome, and two of them ("Clipart" and "Real World") are selected, which contains 8722 images. After randomly splitting all classes, 25 classes serve as the seen classes to train the model, and the remaining 15 and 25 classes are used as two UNSEEN for evaluation. Please refer to SM for more details.

**Evaluation protocols.** Previous approaches [9, 38, 40] usually follow the original setting of [43] and evaluate the models on 600 sampled target tasks (15 test instances per class). In a later study [35], it was suggested that such an evaluation process could potentially introduce high variances. Therefore, we follow the new and more trustworthy evaluation setting to evaluate both baseline models and our approach on 10,000 sampled tasks. We report the mean accuracy (in %) as well as the 95% confidence interval.

**Baseline and embedding adaptation methods.** We re-implement the prototypical network (ProtoNet) [38] as a task-agnostic embedding baseline model. This is known as a very strong approach [7] when the backbone architecture is deep, *i.e.*, residual networks [11]. As suggested by [28], we tune the scalar temperature carefully to scale the logits of both approaches in our re-implementation. As mentioned, we implement the embedding adaptation model with four different function approximators, and denote them as BILSTM, DEEPSETS, GCN, and FEAT (*i.e.* Transformer). The concrete details of each model are included in the SM.

**Backbone pre-training.** Instead of optimizing from scratch, we apply an additional pre-training strategy as suggested in [30, 35]. The backbone network, appended with a **softmax** layer, is trained to classify all SEEN classes with the cross-entropy loss (*e.g.*, 64 classes in the *Mini*ImageNet). The classification performance over the penultimate layer embeddings of sampled 1-shot tasks from the model validation split is evaluated to select the best pre-trained model, whose weights are then used to initialize the embedding function $\mathbf{E}$ in the few-shot learning.

### 5.2. Standard Few-Shot Image Classification

We compare our proposed FEAT method with the instance embedding baselines as well as previous methods on the standard *Mini*ImageNet [43] and *Tiered*ImageNet [33] benchmarks, and then perform detailed analysis on the ablated models. We include additional results with CUB [44] dataset in SM, which shares a similar observation.

#### 5.2.1. Main Results

**Comparison to previous State-of-the-arts.** Table 1 and Table 2 show the results of our method and others on the

**Table 1:** Few-shot classification accuracy on *Mini*ImageNet. ★ CTM [23] and SimpleShot [45] utilize the ResNet-18. (see SM for the full table with confidence intervals and WRN results.).

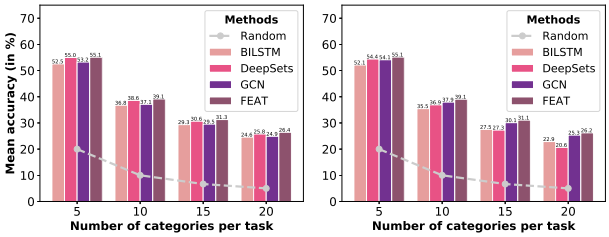
| Setups → | 1-Shot 5-Way | | 5-Shot 5-Way | |
|---|---|---|---|---|
| Backbone → | ConvNet | ResNet | ConvNet | ResNet |
| MatchNet [43] | 43.40 | - | 51.09 | - |
| MAML [9] | 48.70 | - | 63.11 | - |
| ProtoNet [38] | 49.42 | - | 68.20 | - |
| RelationNet [39] | 51.38 | - | 67.07 | - |
| PFA [30] | 54.53 | 59.60 | 67.87 | 73.74 |
| TADAM [28] | - | 58.50 | - | 76.70 |
| MetaOptNet [20] | - | 62.64 | - | 78.63 |
| CTM [23] | - | 64.12 | - | 80.51 |
| SimpleShot [45] | 49.69 | 62.85 | 66.92 | 80.02 |
| **Instance embedding** | | | | |
| ProtoNet | 52.61 | 62.39 | 71.33 | 80.53 |
| **Embedding adaptation** | | | | |
| BILSTM | 52.13 | 63.90 | 69.15 | 80.62 |
| DEEPSETS | 54.41 | 64.14 | 70.96 | 80.93 |
| GCN | 53.25 | 64.50 | 70.59 | 81.65 |
| **FEAT** | **55.15** | **66.78** | **71.61** | **82.05** |

*Mini*ImageNet and *Tiered*ImageNet. First, we observe that the best embedding adaptation method (FEAT) outperforms the instance embedding baseline on both datasets, indicating the effectiveness of learning task-specific embedding space. Meanwhile, the FEAT model performs significantly better than the current state-of-the-art methods on *Mini*ImageNet dataset. On the *Tiered*ImageNet, we observe that the ProtoNet baseline is already better than some previous state-of-the-arts based on the 12-layer ResNet backbone. This might due to the effectiveness of the pre-training stage on the *Tiered*ImageNet as it is larger than *Mini*ImageNet and a fully converged model can be itself very effective. Based on this, all embedding adaptation approaches further improves over ProtoNet almost in all cases, with FEAT achieving the best performances among all approaches. Note that here our pre-training strategy is most similar to the one used in PFA [30], while we further fine-tune the backbone. Temperature scaling of the logits influences the performance a lot when fine-tuning over the pre-trained weights. Additionally, we list some recent methods (SimpleShot [45], and CTM [23]) using different backbone architectures such as ResNet-18 for reference.

**Comparison among the embedding adaptation models.** Among the four embedding adaptation methods, BILSTM in most cases achieves the worst performances and sometimes even performs worse than ProtoNet. This is partially due to the fact that BILSTM can not easily implement the required permutation invariant property (also shown in [49]), which confuses the learning process of embedding adaptation. Secondly, we find that DEEPSETS and GCN have the ability to adapt discriminative task-specific embeddings but do not achieve consistent performance improvement over the baseline ProtoNet especially on *Mini*ImageNet with the ConvNet backbone. A potential explanation is that, such models when jointly learned with the backbone model, can make the optimization process more difficult, which leads to the varying final performances. In contrast, we observe that FEAT can consistently improve ProtoNet and other embedding adaptation approaches in all cases, without additional bells and whistles. It shows that the Transformer as a set-to-set function can implement rich interactions between instances, which provides its high expressiveness to model the embedding adaptation process.

**Table 2:** Few-shot classification accuracy and 95% confidence interval on *Tiered*ImageNet with the ResNet backbone.

| Setups → | 1-Shot 5-Way | 5-Shot 5-Way |
|---|---|---|
| ProtoNet [38] | 53.31 ± 0.89 | 72.69 ± 0.74 |
| RelationNet [39] | 54.48 ± 0.93 | 71.32 ± 0.78 |
| MetaOptNet [20] | 65.99 ± 0.72 | 81.56 ± 0.63 |
| CTM [23] | 68.41 ± 0.39 | 84.28 ± 1.73 |
| SimpleShot [45] | 69.09 ± 0.22 | 84.58 ± 0.16 |
| **Instance embedding** | | |
| ProtoNet | 68.23 ± 0.23 | 84.03 ± 0.16 |
| **Embedding adaptation** | | |
| BILSTM | 68.14 ± 0.23 | 84.23 ± 0.16 |
| DEEPSETS | 68.59 ± 0.24 | 84.36 ± 0.16 |
| GCN | 68.20 ± 0.23 | 84.64 ± 0.16 |
| **FEAT** | **70.80** ± **0.23** | **84.79** ± **0.16** |




(*a*) **Way Interpolation** (*b*) **Way Extrapolation**

**Figure 3: Interpolation** and **Extrapolation** of few-shot tasks from the "way" perspective. First, We train various embedding adaptation models on 1-shot 20-way (a) or 5-way (b) classification tasks and evaluate models on unseen tasks with different number of classes ($N$={*5, 10, 15, 20*}). It shows that FEAT is superior in terms of way interpolation and extrapolation ability.

**Table 3:** Number of parameters introduced by each set-to-set function in additional to the backbone's parameters.

| | BILSTM | DEEPSETS | GCN | FEAT |
|---|---|---|---|---|
| ConvNet | 25K | 82K | 33K | 16K |
| ResNet | 2.5M | 8.2M | 3.3M | 1.6M |

**Interpolation and extrapolation of classification ways.** Next, we study different set-to-set functions on their capability of interpolating and extrapolating across the number of classification ways. To do so, we train each variant of embedding adaptation functions with both 1-shot 20-way and 1-shot 5-way tasks, and measure the performance change as a function to the number of categories in the test time. We report the mean accuracies evaluated on few-shot classification with $N = \{5, 10, 15, 20\}$ classes, and show results in Figure 3. Surprisingly, we observe that FEAT *achieves almost the same numerical performances in both extrapolation and interpolation scenarios*, which further displays its strong capability of learning the set-to-set transformation. Meanwhile, we observe that DEEPSETS works well with interpolation but fails with extrapolation as its performance drops significantly with the larger $N$. In contrast, GCN achieves strong extrapolation performances but does not work as effectively in interpolation. BILSTM performs the worst in both cases, as it is by design not permutation invariant and may have fitted an arbitrary dependency between instances.

**Parameter efficiency.** Table 3 shows the number of additional parameters each set-to-set function has introduced. From this, we observe that with both ConvNet and ResNet backbones, FEAT has the smallest number of parameters compared with all other approaches while achieving best performances from various aspects (as results discussed above), which highlights its high parameter efficiency.

All above, we conclude that: 1) learning embedding adaptation with a set-to-set model is very effective in modeling task-specific embeddings for few-shot learning 2) FEAT is the most parameter-efficient function approximater that achieves the best empirical performances, together with nice permutation invariant property and strong interpolation/extrapolation capability over the classification way.

### 5.2.2. Ablation Studies

We analyze FEAT and its ablated variants on the *Mini*ImageNet dataset with ConvNet backbone.

**How does the embedding adaptation looks like qualitatively?** We sample four few-shot learning tasks and learn a principal component analysis (PCA) model (that projects embeddings into 2-D space) using the instance embeddings of the test data. We then apply this learned PCA projection to both the support set's pre-adapted and post-adapted embeddings. The results are shown in Figure 1 (the beginning of the paper). In three out of four examples, post-adaptation embeddings of FEAT improve over the pre-adaption embeddings. Interestingly, we found that the embedding adaptation step of FEAT has the tendency of pushing the support embeddings apart from the clutter, such that they can better fit the test data of its categories. In the negative example where post-adaptation degenerates the performances,

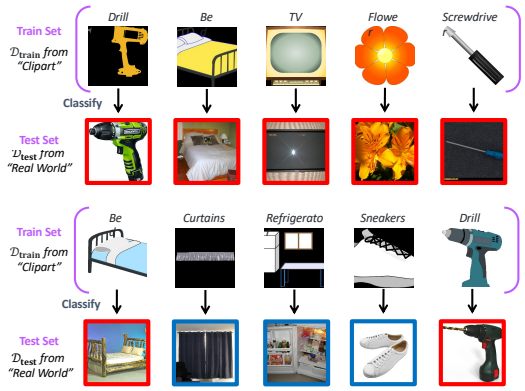



**Figure 4:** Qualitative results of **few-shot domain-generalization** for FEAT. Correctly classified examples are shown in **red** boxes and incorrectly ones are shown in **blue** boxes. We visualize one task that FEAT succeeds (top) and one that fails (bottom).

we observe that the embedding adaptation step has pushed two support embeddings "Golden Retriever" and "Lion" too close to each other. It has qualitatively shown that the adaptation is crucial to obtain superior performances and helps to contrast against task-agnostic embeddings.

## 5.3. Extended Few-Shot Learning Tasks

In this section, we evaluate FEAT on 3 different few-shot learning tasks. Specifically, cross-domain FSL, transductive FSL [25, 33], and generalized FSL [6]. We overview the setups briefly and please refer to SM for details.

**FS Domain Generalization** assumes that examples in UNSEEN support and test set can come from the different domains, *e.g.*, sampled from different distributions [8, 14]. The example of this task can be found in Figure 4. It requires a model to recognize the intrinsic property than texture of objects, and is de facto analogical recognition.

**Transductive FSL.** The key difference between standard and transductive FSL is whether test instances arrive one at a time or all simultaneously. The latter setup allows the structure of unlabeled test instances to be utilized. Therefore, the prediction would depend on both the training (support) instances and all the available test instances in the target task from UNSEEN categories.

**Generalized FSL.** Prior works assumed the test instances coming from unseen classes only. Different from them, the *generalized FSL* setting considers test instances from both SEEN and UNSEEN classes [32]. In other words, during the model evaluation, while support instances all come from $\mathcal{U}$, the test instances come from $\mathcal{S} \cup \mathcal{U}$, and the classifier is required to predict on both SEEN and UNSEEN categories. .

### 5.3.1. Few-Shot Domain Generalization

We show that FEAT learns to adapt *the intrinsic structure of tasks*, and **generalizes across domains**, *i.e.*, predicting

| | C → C | C → R |
|---|---|---|
| Supervised | 34.38±0.16 | 29.49±0.16 |
| ProtoNet | 35.51±0.16 | 29.47±0.16 |
| FEAT | **36.83**±0.17 | **30.89**±0.17 |

**(a) Few-shot domain generalization**

| | 1-Shot | 5-Shot |
|---|---|---|
| TPN [25] | 55.51 | 69.86 |
| TEAM [29] | 56.57 | 72.04 |
| FEAT | **57.04** ± 0.20 | **72.89** ± 0.16 |

**(b) Transductive few-shot learning**

| | SEEN | UNSEEN | COMBINED |
|---|---|---|---|
| Random | 1.56 ±0.00 | 20.00±0.00 | 1.45±0.00 |
| ProtoNet | 41.73±0.03 | 48.64±0.20 | 35.69±0.03 |
| FEAT | **43.94**±0.03 | **49.72**±0.20 | **40.50**±0.03 |

**(c) Generalized few-shot learning**

**Table 4:** We evaluate our model on three additional few-shot learning tasks: **(a)** Few-shot domain generalization, **(b)** Transductive few-shot learning, and **(c)** Generalized few-shot learning. We observe that FEAT consistently outperform all previous methods or baselines.

test instances even when the visual appearance is changed.

**Setups.** We train the FSL model in the standard domain and evaluate with cross-domain tasks, where the $N$-categories are aligned but domains are different. In detail, a model is trained on tasks from the "Clipart" domain of OfficeHome dataset [42], then the model is required to generalize to both "Clipart (**C**)" and"Real World (**R**)" test instances. In other words, we need to classify complex real images by seeing only a few sketches (Figure 4 gives an overview of data).

**Results.** Table 4 (a) gives the quantitative results and Figure 4 qualitatively examines it. Here, the "supervised" denotes a model trained with the standard classification strategy and then its penultimate layer's output feature is used as the nearest neighbor classifier. We observe that ProtoNet can outperform this baseline on tasks when evaluating instances from "Clipart" but not ones from "real world". However, FEAT improves over "real world" few-shot classification even only seeing the support data from "Clipart".

### 5.3.2. Transductive Few-Shot Learning

We show that without additional efforts in modeling, FEAT outperforms existing methods in transductive FSL.

**Setups.** We further study this semi-supervised learning setting to see how well FEAT can incorporate test instances into joint embedding adaptation. Specifically, we use the unlabeled test instances to augment the key and value sets of Transformer (refer to SM for details), so that the embedding adaptation takes relationship of all test instances into consideration. We evaluate this setting on the transductive protocol of *Mini*ImageNet [33]. With the adapted embedding, FEAT makes predictions based on Semi-ProtoNet [33].

**Results.** We compare with two previous approaches, TPN [25] and TEAM [29]. The results are shown in Table 4 (b). We observe that FEAT improves its standard FSL performance (refer to Table 1) and also outperforms previous semi-supervised approaches by a margin.

### 5.3.3. Generalized Few-Shot Learning

We show that FEAT performs well on generalized few-shot classification of both SEEN and UNSEEN classes.

**Setups.** In this scenario, we evaluate not only on classifying test instances from a $N$-way $M$-shot task from UNSEEN set $\mathcal{U}$, but also on all available SEEN classes from $\mathcal{S}$. To do so, we hold out 150 instances from each of the 64 seen classes in *Mini*ImageNet for validation and evaluation. Next, given a 1-shot 5-way training set $\mathcal{D}_{\mathbf{train}}$, we consider three evaluation protocols based on different class sets [6]: UNSEEN measures the mean accuracy on test instances only from $\mathcal{U}$ (5-Way few-shot classification); SEEN measures the mean accuracy on test instances only from $\mathcal{S}$ (64-Way classification); COMBINED measures the mean accuracy on test instances from $\mathcal{S} \cup \mathcal{U}$ (69-Way mixed classification).

**Results.** The results can be found in Table 4 (c). We observe that again FEAT outperforms baseline ProtoNet. To calibrate the prediction score on SEEN and UNSEEN classes [6, 46], we select a constant seen/unseen class probability over the validation set, and subtract this calibration factor from seen classes' prediction score. Then we take the prediction with maximum score value after calibration.

## 6. Discussion

A common embedding space fails to tailor discriminative visual knowledge for a target task especially when there are a few labeled training data. We propose to do embedding adaptation with a set-to-set function and instantiate it with transformer (FEAT), which customizes task-specific embedding spaces via a self-attention architecture. The adapted embedding space leverages the relationship between target task training instances, which leads to discriminative instance representations. FEAT achieves the state-of-the-art performance on benchmarks, and its superiority can generalize to tasks like cross-domain, transductive, and generalized few-shot classifications.

**Acknowledgments.** This work is partially supported by The National Key R&D Program of China (2018YFB1004300), DARPA# FA8750-18-2-0117, NSF IIS-1065243, 1451412, 1513966/ 1632803/1833137, 1208500, CCF-1139148, a Google Research Award, an Alfred P. Sloan Research Fellowship, ARO# W911NF-12-1-0241 and W911NF-15-1-0484, China Scholarship Council (CSC), NSFC (61773198, 61773198, 61632004), and NSFC-NRF joint research project 61861146001.

## References


[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for attribute-based classification. In *CVPR*, pages 819–826, 2013. 2

[2] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, pages 3981–3989. 2016. 2

[3] A. Antoniou, H. Edwards, and A. J. Storkey. How to train your MAML. In *ICLR*, 2019. 2

[4] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. In *CVPR*, pages 5327–5336, 2016. 2

[5] S. Changpinyo, W.-L. Chao, and F. Sha. Predicting visual exemplars of unseen classes for zero-shot learning. In *ICCV*, pages 3496–3505, 2017. 2

[6] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV*, pages 52–68, 2016. 7, 8

[7] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *ICLR*, 2019. 5

[8] N. Dong and E. P. Xing. Domain adaption in one-shot learning. In *ECML PKDD*, pages 573–588, 2018. 7

[9] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017. 1, 2, 3, 5, 6

[10] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. M. F. Moura. Few-shot human motion prediction via meta-learning. In *ECCV*, pages 441–459, 2018. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2, 4, 5

[13] K. Hsu, S. Levine, and C. Finn. Unsupervised learning via meta-learning. In *ICLR*, 2019. 2

[14] B. Kang and J. Feng. Transferable meta learning across domains. In *UAI*, pages 177–187, 2018. 7

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[16] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 2, 4, 5

[17] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015. 2

[18] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learning of simple visual concepts. In *CogSci*, 2011. 1

[19] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 1

[20] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pages 10657–10665, 2019. 2, 5, 6

[21] Y. Lee and S. Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, pages 2933–2942, 2018. 2

[22] F.-F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006. 1

[23] H. Li, D. Eigen, S. Dodge, M. Zeiler, and X. Wang. Finding task-relevant features for few-shot learning by category traversal. In *CVPR*, pages 1–10, 2019. 2, 6

[24] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017. 2, 4

[25] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *ICLR*, 2019. 7, 8

[26] L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein. Learning unsupervised learning rules. *CoRR*, abs/1804.00222, 2018. 2

[27] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. 2

[28] B. N. Oreshkin, P. R. López, and A. Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, pages 719–729. 2018. 5, 6

[29] L. Qiao, Y. Shi, J. Li, Y. Wang, T. Huang, and Y. Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *ICCV*, pages 3603–3612, 2019. 8

[30] S. Qiao, C. Liu, W. Shen, and A. L. Yuille. Few-shot image recognition by predicting parameters from activations. In *CVPR*, pages 7229–7238, 2018. 2, 5, 6

[31] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 2, 5

[32] M. Ren, R. Liao, E. Fetaya, and R. S. Zemel. Incremental few-shot learning with attention attractor networks. *CoRR*, abs/1810.07218, 2018. 7

[33] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018. 5, 7, 8

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[35] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019. 1, 2, 5

[36] V. G. Satorras and J. B. Estrach. Few-shot learning with graph neural networks. In *ICLR*, 2018. 2, 4

[37] T. R. Scott, K. Ridgeway, and M. C. Mozer. Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. In *NeurIPS*, pages 76–85. 2018. 2

[38] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4080–4090. 2017. 1, 2, 3, 4, 5, 6

[39] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208, 2018. 2, 6

[40] E. Triantafillou, R. S. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. In *NeurIPS*, pages 2252–2262. 2017. 1, 2, 5

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, pages 6000–6010. 2017. 2, 4, 5

[42] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, pages 5385–5394, 2017. 5, 8

[43] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NeurIPS*, pages 3630–3638. 2016. 1, 2, 3, 4, 5, 6

[44] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 5

[45] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019. 6

[46] Y.-X. Wang, R. B. Girshick, M. Hebert, and B. Hariharan. Low-shot learning from imaginary data. In *CVPR*, pages 7278–7286, 2018. 8

[47] X.-S. Wei, P. Wang, L. Liu, C. Shen, and J. Wu. Piecewise classifier mappings: Learning fine-grained learners for novel categories with few examples. *TIP*, 28(12):6116–6125, 2019. 2

[48] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 5

[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *NeurIPS*, pages 3394–3404. 2017. 2, 4, 5, 6