XtarNet: Learning to Extract Task-Adaptive Representation for Incremental Few-Shot Learning

Sung Whan Yoon [1] **Do-Yeon Kim** [2] **Jun Seo** [2] **Jaekyun Moon** [2]

Abstract

Learning novel concepts while preserving prior knowledge is a long-standing challenge in machine learning. The challenge gets greater when a novel task is given with only a few labeled examples, a problem known as *incremental few-shot learning*. We propose *XtarNet*, which learns to extract task-adaptive representation (TAR) for facilitating incremental few-shot learning. The method utilizes a backbone network pretrained on a set of base categories while also employing additional modules that are meta-trained across episodes. Given a new task, the novel feature extracted from the meta-trained modules is mixed with the base feature obtained from the pretrained model. The process of combining two different features provides TAR and is also controlled by meta-trained modules. The TAR contains effective information for classifying both novel and base categories. The base and novel classifiers quickly adapt to a given task by utilizing the TAR. Experiments on standard image datasets indicate that XtarNet achieves state-of-the-art incremental few-shot learning performance. The concept of TAR can also be used in conjunction with existing incremental few-shot learning methods; extensive simulation results in fact show that applying TAR enhances the known methods significantly.

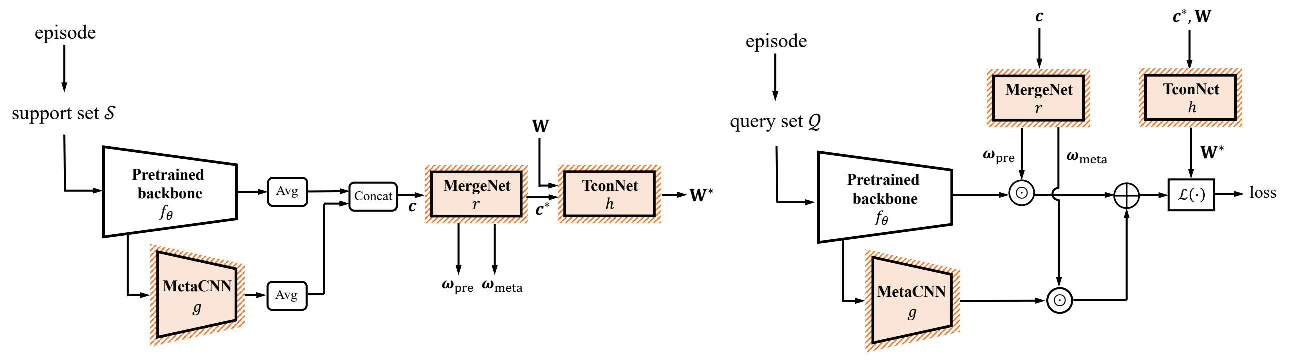
1. Introduction

Humans can quickly learn novel concepts from limited experience. In contrast, deep learning usually requires a massive amount of training data to learn each novel task. Unfortunately, gathering sufficient training samples in many applications can be highly expensive and impractical. To tackle a related problem in computer vision, many researchers have dedicated to develop few-shot learning algorithms, which aim to learn generalized models for classifying unseen categories with only a few labeled images. A popular way to develop few-shot learners is to adopt meta-training in *episodic* form (Vinyals et al., 2016), where the learners are exposed to widely-varying episodes one by one, each time with a few-labeled images. Built upon the episodic training, successful few-shot learning algorithms attempt to build proper inductive bias over varying tasks while employing effective task-dependent adaptation for each task (Snell et al., 2017; Finn et al., 2017; Mishra et al., 2017; Oreshkin et al., 2018; Yoon et al., 2019).

While well-known few-shot learning methods focus mainly on classifying novel categories, a recent line of works attempts to handle novel categories while retaining the ability to classify a set of original base categories (Gidaris & Komodakis, 2018; Ren et al., 2019). This more recent problem is referred to as incremental few-shot learning. Here, based on a prepared model which is trained on a set of original base categories, extra novel categories should be quickly learned within a few shots. To this end, incremental few-shot learners strive to build a generalized model while preventing catastrophic forgetting of the base categories. Achieving this objective is extremely arduous; simple extensions of existing few-shot learning algorithms to the setting of incremental few-shot learning does not work. Built upon a pretrained backbone, the recent methods of (Gidaris & Komodakis, 2018; Ren et al., 2019) focus on learning to learn a set of new weights for novel categories that can balance the classification performance on base and novel categories. The attention-based method of (Gidaris & Komodakis, 2018) learns to leverage the past knowledge by generating novel class weights with a form of attention over the pretrained classification weights for base categories. A more recent method of (Ren et al., 2019) called Attention Attractor Network employs an inner loop of optimization for generating a novel classifier for each episode. To be specific, meta-learned modules compute an attractor vector, and then a regularization term based on the distance between the attractor and novel classification weights is used in the inner loop optimization. The prior methods which learn to build novel classifier show substantial performance gains over the

[1]School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan, Korea [2]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. Correspondence to: Sung Whan Yoon <shyoon8@unist.ac.kr>.

Work in Progress.


episode
support set $\mathcal{S}$
Pretrained backbone f_θ
MetaCNN g
Avg
Avg
Concat
$\boldsymbol{c}$
MergeNet r
$\boldsymbol{c}^*$
$\mathbf{W}$
TconNet h
$\mathbf{W}^*$
$\boldsymbol{\omega}_{\text{pre}}$
$\boldsymbol{\omega}_{\text{meta}}$
episode
query set $\mathcal{Q}$
Pretrained backbone f_θ
MetaCNN g
$\boldsymbol{c}$
MergeNet r
$\boldsymbol{c}^*, \mathbf{W}$
TconNet h
$\boldsymbol{\omega}_{\text{pre}}$
$\boldsymbol{\omega}_{\text{meta}}$
$\mathbf{W}^*$
$\mathcal{L}(\cdot)$
loss

(a) Processing of a support set (b) Processing of a query set

Figure 1. XtarNet processing of a given episode.

baseline approach that uses per-class averages of embedded support samples as novel classification weights. However, these algorithms focus on extracting a *fixed* representation from the pretrained backbone, rather than exploring novel representations that could be task-adaptive.

Our motivation here is to pursue novel knowledge from new experience and merge it with prior knowledge learned from previous experience. In this paper, we propose *XtarNet* which learns to construct novel representation with informative features to classify both base and novel categories. In addition to a pretrained feature extractor, our method employs additional modules that are meta-trained across varying episodes to accomplish the following three objectives: design of a novel feature extractor that can be trained easily across different tasks, construction of *task-adaptive representation* (TAR) by mixing the novel and base features together, and provision of efficient task-conditioning of base and novel classifiers by utilizing the TAR.

Experiments on incremental few-shot classification tasks with *mini*ImageNet and *tiered*ImageNet show that XtarNet achieves the state-of-the-art performance. The proposed TAR idea can be used in conjunction with different few-shot learning methods, improving incremental few-shot classification accuracies of existing learners.

2. Proposed Model

We follow the incremental few-shot learning setting proposed in (Gidaris & Komodakis, 2018; Ren et al., 2019). First, a backbone network is pretrained on a set of base categories. Built upon the pretrained model, meta-training is done with episodic form of training. For each episode of incremental few-shot learning, a few labeled samples (i.e., support samples) from novel categories are given to the model. Based on the support set, the model aims to classify query samples from both base and novel categories.

Our XtarNet utilizes three different meta-learnable modules in addition to a backbone network. Fig. 1a describes how XtarNet processes the support set $\mathcal{S}$ of a given episode. The backbone feature extractor f_θ is prepared through a regular supervised learning on a training dataset for base categories. The classification weight vector set $\mathbf{W}$ consist of base classifier weight vectors, which are also pretrained on base categories, as well as novel classifier weight vectors, which vary with each training episode. *MetaCNN* g is a small convolutional neural netwok (CNN) module that takes as input an intermediate layer output of f_θ. The output of g represents the *novel feature*. Per-class average features are collected at the outputs of both backbone and MetaCNN. The backbone network output represents the *base feature*. *MergeNet* r, consisting of two relatively small fully-connected networks, creates a mixture of the base and novel features. This mixture is what we call the task-adaptive representation (TAR) of the given input image. *TconNet* h, based on another set of small fully-connected networks, provides task-conditioning on the classification weights $\mathbf{W}$ by utilizing the TAR.

Fig. 1b depicts processing of a query set and the learning process. Once the support set is processed, the mixture weight vectors, $\boldsymbol{\omega}_{\text{pre}}$ and $\boldsymbol{\omega}_{\text{meta}}$, as well as the conditioned weights $\mathbf{W}^*$ become available to be used during the query processing. The given query image is processed again through both backbone and MetaCNN, and the output features are combined via the given weights as shown. Cross entropy loss is computed using the classifier $\mathbf{W}^*$, which in turn is used to update the learnable parameters g, r and h. Note that the mixture feature vectors $\mathbf{c}$ and $\mathbf{c}^*$ and the weight $\mathbf{W}$ used here were also from the support set processing stage.

During meta-training, each episode is presented one at a time, with the above-mentioned support set and query set processing repeated each time. In the proposed model structure, MetaCNN and the neural networks within the MergeNet and TconNet modules attempt to learn a good inductive bias through episodic meta-training, while the TAR mixture weight vectors $\boldsymbol{\omega}_{\text{pre}}$ and $\boldsymbol{\omega}_{\text{meta}}$ and the classification weights $\mathbf{W}^*$, which are generated anew given the support set of each new episode, provide quick task-conditioning for inference. Details of the training procedures and module operations are given in the following subsections.

2.1. Training Procedures

Dataset Splits: $\mathcal{D}_{\text{base}}$ denotes the dataset used in pretraining, and $\mathcal{D}_{\text{base/train}}$, $\mathcal{D}_{\text{base/val}}$ and $\mathcal{D}_{\text{base/test}}$ are training, validation and test sets of $\mathcal{D}_{\text{base}}$, respectively. For incremental setting, a dataset $\mathcal{D}_{\text{novel}}$ is for providing novel categories. The dataset consists of three splits of $\mathcal{D}_{\text{novel/train}}$, $\mathcal{D}_{\text{novel/val}}$ and $\mathcal{D}_{\text{novel/test}}$, containing disjoint categories.

Pretraining phase: On a data split $\mathcal{D}_{\text{base/train}}$ with a fixed set of N_b base classes, we train the embedding network f_θ and the base classifier weights of $\mathbf{W}$.

Meta-training phase: Meta-training is done on two data splits $\mathcal{D}_{\text{base/train}}$ and $\mathcal{D}_{\text{novel/train}}$. For each episode, N novel classes are randomly chosen from the training set $\mathcal{D}_{\text{novel/train}}$. Then we relabel these categories with new labels from N_b+1 to $N_b + N$. Also, K per-class data samples are randomly chosen from the set of novel categories for constructing a support set $\mathcal{S} = \{(\mathbf{x}_j^S, y_j^S)\}_{j=1}^{N\times K}$. The query set $\mathcal{Q}$ consists of samples not only from the novel but also the base categories. To construct a novel query set $\mathcal{Q}_{\text{novel}}$, additional samples are randomly chosen from the novel categories. For base query samples, we randomly select N base categories from $\mathcal{D}_{\text{base/train}}$, and then pick samples for a base query set $\mathcal{Q}_{\text{base}}$. The overall query set is constructed as a union of sets; $\mathcal{Q} = \mathcal{Q}_{\text{base}} \cup \mathcal{Q}_{\text{novel}}$. For each episode, learnable parameters are updated by computing loss during classification of queries in $\mathcal{Q}$.

Test Phase: Upon deployment, all learnable parameters are fixed. For each test episode, N novel categories are randomly chosen from split $\mathcal{D}_{\text{novel/test}}$ (or split $\mathcal{D}_{\text{novel/val}}$ for validation) which consists of unseen categories during both the pretraining and the meta-training phases. For base categories, test samples are chosen from test split $\mathcal{D}_{\text{base/test}}$ (or split $\mathcal{D}_{\text{base/val}}$ for validation).

2.2. MetaCNN for Extracting Novel Feature

For a given input $\mathbf{x}$ from the support set, let $a_\theta(\mathbf{x})$ denote the intermediate layer output of the pretrained backbone f_θ and be used as an input to MetaCNN. This way, MetaCNN extracts a different high-level feature vector $g(a_\theta(\mathbf{x}))$. Let us call the vector $f_\theta(\mathbf{x}) \in \mathbb{R}^D$ and $g(a_\theta(\mathbf{x})) \in \mathbb{R}^D$ the base feature and novel feature, respectively. D is the length of the features.

2.3. MergeNet for Combining Base and Novel Features

MergeNet r is a meta-trained module for obtaining task-specific mixture weight vectors $\boldsymbol{\omega}_{\text{pre}}$ and $\boldsymbol{\omega}_{\text{meta}}$, which are used to combine the base and novel feature vectors. For each support sample in a given $\mathcal{S}$, the base and novel features are extracted by the pretrained backbone and MetaCNN, and then concatenated. Let $[f_\theta(\mathbf{x}), g(a_\theta(\mathbf{x}))]$ denote this concatenated feature vector. Afterwards, a task-representation vector $\mathbf{c}$ is computed by averaging all concatenated vectors of the support examples, i.e., $\mathbf{c} = \frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}\in\mathcal{S}}[f_\theta(\mathbf{x}), g(a_\theta(\mathbf{x}))]$. MergeNet consists of two separate fully-connected networks r_{pre} and r_{meta}, which compute $\boldsymbol{\omega}_{\text{pre}}$ and $\boldsymbol{\omega}_{\text{meta}}$ from $\mathbf{c}$, respectively: $\boldsymbol{\omega}_{\text{pre}} = r_{\text{pre}}(\mathbf{c})$ and $\boldsymbol{\omega}_{\text{meta}} = r_{\text{meta}}(\mathbf{c})$.

After preparing $\boldsymbol{\omega}_{\text{pre}}$ and $\boldsymbol{\omega}_{\text{meta}}$, a combined feature vector is obtained by the summation of element-wise products between these mixture weight vectors and feature vectors: $\boldsymbol{\omega}_{\text{pre}} \odot f_\theta(\mathbf{x}) + \boldsymbol{\omega}_{\text{meta}} \odot g(a_\theta(\mathbf{x}))$, where $\odot$ is an element-wise product operation. For each novel class k in the given support set, the per-class average $\mathbf{c}_k^*$ of the combine feature vectors is calculated as

$$\mathbf{c}_k^* = \frac{1}{|\mathcal{S}_k|}\sum_{\mathbf{x}\in\mathcal{S}_k}[\boldsymbol{\omega}_{\text{pre}} \odot f_\theta(\mathbf{x}) + \boldsymbol{\omega}_{\text{meta}} \odot g(a_\theta(\mathbf{x}))], \quad (1)$$

where $N_b + 1 \le k \le N_b + N$ and $\mathcal{S}_k$ is the set of support samples from novel class k. For each episode, MergeNet learns to build the mixture weights which are used to construct the TAR.

2.4. TconNet for Conditioning Classifiers

TconNet h consists of two meta-learned modules for task-conditioning of the classifier. One module conditions the pretrained base classification weights $\{\mathbf{w}_i\}_{i=1}^{N_b} \in \mathbf{W}$. First, the average of the combined feature vectors of support samples is obtained: $\mathbf{c}^* = \frac{1}{N}\sum_{k=N_b+1}^{N_b+N}\mathbf{c}_k^*$. With two separate fully-connected networks h_γ and h_β, two conditioning vectors are obtained from $\mathbf{c}^*$: $h_\gamma(\mathbf{c}^*)$ and $h_\beta(\mathbf{c}^*)$. The task-conditioned base classification weights $\{\mathbf{w}_i^*\}_{i=1}^{N_b}$ are obtained by utilizing these vectors for scaling and biasing:

$$\mathbf{w}_i^* = (\mathbb{1} + h_\gamma(\mathbf{c}^*)) \odot \mathbf{w}_i + h_\beta(\mathbf{c}^*) \quad (2)$$

where $\mathbb{1}$ is a all 1's vector with length D. The weights $\{\mathbf{w}_i^*\}_{i=1}^{N_b}$ are used as the base classifier. We now consider the initial classification weights $\{\mathbf{w}_i\}_{i=N_b+1}^{N_b+N} \in \mathbf{W}$ for the novel categories.

Preparing novel classifier: Perhaps the simplest novel classifier setting is that of the Imprint method of (Qi et al., 2018); simply set the per-class feature averages as the novel classification weights: $\{\mathbf{c}_k^*\}_{k=N_b+1}^{N_b+N}$ is set to be $\{\mathbf{w}_i\}_{i=N_b+1}^{N_b+N}$. The initial novel classification weights can also be computed by LwoF, the attention-based method of (Gidaris & Komodakis, 2018). This approach employs a meta-learned attention-based weight generator for the novel classifier. When used with our XtarNet, the weight generator is learned along with other modules within XtarNet during meta-training. Yet another way is to utilize TapNet of (Yoon et al., 2019), which is a relatively simple but effective few-shot learning method. TapNet utilizes meta-learned per-class reference vectors for classification in a task-adaptive projection space. The reference vectors $\mathbf{\Phi} = \{\boldsymbol{\phi}_n\}_{n=1}^{N}$ of TapNet are learned across episodes while the projection space $\mathbf{M}$ is computed anew specific to each episode in such as way that class prototypes and the matching reference vectors align in $\mathbf{M}$. After obtaining per-class averages of embedded features, $\{\mathbf{c}_n\}_{n=1}^{N}$, space $\mathbf{M}$ is constructed such that $\{\boldsymbol{\phi}_n\}_{n=1}^{N}$ and $\{\mathbf{c}_n\}_{n=1}^{N}$ align in $\mathbf{M}$. More specifically, the errors defined as

$$\boldsymbol{\epsilon}_n = \frac{\boldsymbol{\phi}_n}{\|\boldsymbol{\phi}_n\|} - \frac{\mathbf{c}_n}{\|\mathbf{c}_n\|} \tag{3}$$

are forced to zero when projected in $\mathbf{M}$, i.e., $\boldsymbol{\epsilon}_n\mathbf{M} = \mathbf{0}$ for all class n. Such an $\mathbf{M}$ can be found by taking the null-space of the errors by using singular value decomposition (SVD). When XtarNet is used in conjunction with TapNet, we simply set $\{\mathbf{w}_i\}_{i=N_b+1}^{N_b+N} \leftarrow \{\boldsymbol{\phi}_n\}_{n=1}^{N}$ and also utilize task-adaptive projection to perform classification.

Adaptation of novel classifier: After the initial novel classifier $\{\mathbf{w}_i\}_{i=N_b+1}^{N_b+N}$ is obtained, another fully-connected network h_λ of TconNet learns to separate novel classifier weights from the task-conditioned base classifier weights by introducing bias terms. First, a correlation vector $\boldsymbol{\sigma}_k$ is calculated for each novel class k, whose i^{th} element σ_k^i is the inner-product similarity between the combined per-class average feature $\mathbf{c}_k^*$ and the base classification weight, i.e., $\sigma_k^i = \mathbf{c}_k^* \cdot \mathbf{w}_i^*$ for all $i = 1, \cdots, N_b$. For each novel class k, the module takes the correlation vector $\boldsymbol{\sigma}_k \in \mathbb{R}^{N_b}$ as the input to calculate: $\boldsymbol{\lambda}_k = h_\lambda(\boldsymbol{\sigma}_k) \in \mathbb{R}^{N_b}$. Then the output vector is utilized to modify $\mathbf{w}_i$ by introducing a bias term, which is a weighted sum of the conditioned base classification weights $\{\mathbf{w}_j^*\}_{j=1}^{N_b}$:

$$\mathbf{w}_i^* = \mathbf{w}_i - \sum_{j=1}^{N_b} \frac{\exp(\lambda_i^j)}{\sum_l \exp(\lambda_i^l)} \mathbf{w}_j^* \tag{4}$$

for all $N_b + 1 \le i \le N_b + N$. λ_k^i is i^{th} element of $\boldsymbol{\lambda}_k$. This completes the overall set of the task-conditioned weight vectors $\mathbf{W}^* = \{\mathbf{w}_i^*\}_{i=1}^{N_b+N}$ for classifying queries.

For best results, our XtarNet utilizes the task-adaptive projection space of TapNet. The projection space $\mathbf{M}$ is computed such that the per-class averages of the combined features $\{\mathbf{c}_k^*\}_{k=N_b+1}^{N_b+N}$ and the conditioned novel classifier weights of $\mathbf{W}^*$ coincide in $\mathbf{M}$.

For a given query $\mathbf{x} \in \mathcal{Q}$, the Euclidean distance $D_i(\mathbf{x})$ between the projected feature of query and the projected classifier weights is used in classification:

$$D_i(\mathbf{x}) = d\Big(\mathbf{w}_i^*\mathbf{M}, [\boldsymbol{\omega}_{\text{pre}} \odot f_\theta(\mathbf{x}) + \boldsymbol{\omega}_{\text{meta}} \odot g(a_\theta(\mathbf{x}))]\mathbf{M}\Big), \tag{5}$$

Finally, classification of the query $\mathbf{x}$ is carried out based on the posterior probabilities:

$$p(y = i|\mathbf{x}) = \frac{\exp(-D_i(\mathbf{x}))}{\sum_l \exp(-D_l(\mathbf{x}))}, \tag{6}$$

for all $i = 1, \cdots, N_b + N$.

3. Related Work

Few-Shot Learning: Few-shot learning attempts to train models for novel tasks with only a few labeled samples. The metric-learning-based approach is one of the most popular branches of few-shot learning. The key idea of metric-based few-shot learners is to train the feature embedding so that the embedded samples belong to the same class are located closely. Matching Networks of (Vinyals et al., 2016) learn the embedding space and the query samples are classified based on the similarity scores computed by the distance between the few labeled support samples and query samples. Note that two different embedding spaces can be learned to process the support set and query set. Prototypical Networks of (Snell et al., 2017) also rely on the embedding network where the samples from the same class are clustered together. Instead of computing similarity between samples, Prototypical Networks utilize per-class averages as the prototypes. Classification of queries is done by finding the nearest prototype in the embedding space.

Task-Conditioning: Recently, few-shot learning algorithms utilize explicit task-conditioning and achieve significant performance gain. TADAM of (Oreshkin et al., 2018) offers a task-conditioning method based on the general-purpose conditioning of FiLM proposed in (Perez et al., 2018). Like FiLM, TADAM performs affine transformation on the middle level feature. The scaling and shifting parameters are generated from task representation, which is the averaged feature of support samples. An additional task-conditioning module is meta-learned to generate the conditioning parameters, and auxiliary co-training is adopted for efficient meta-learning. The component module TconNet of our XtarNet also provides task-conditioning in a form of affine transformation to adjust the base classifier. TapNet of (Yoon et al., 2019) proposed another type of task-conditioning based on feature projection. Additional meta-learned reference vectors are utilized in classification. When a task is given, a projection space is constructed such that the class prototypes and the reference vector closely align there. Classification is done in the projection space by measuring distance between projected queries and references. These prior works and our method are common in the sense that explicit task-conditioning is employed, but our XtarNet is unique in that task-conditioning is made possible by creating a mixture of the base and novel features that depends on the given task.

Incremental Learning: Learning from the continuously arriving data while preserving learned knowledge is called incremental learning. The key to incremental learning is preventing catastrophic forgetting (McCloskey & Cohen, 1989; McClelland et al., 1995), which refers to the situation where the model forgets learned prior knowledge. One way to prevent catastrophic forgetting is using memory (Rebuffi et al., 2017; Nguyen et al., 2018). The memory-based model stores prior training data explicitly and use it for training again at the appropriate moment. Generative models can also be used to prevent forgetting by generating previously learned data when data is needed (Shin et al., 2017; Kemker & Kanan, 2018). Regularization can be used for learning from new data while preventing catastrophic forgetting, without replaying previous data (Kirkpatrick et al., 2017). Learning without forgetting is another way to prevent forgetting (Li & Hoiem, 2017). The information about prior data is not stored, and feature extracted from new data using an old model is utilized to preserve prior knowledge. Our XtarNet is closest to the Learning without forgetting method of (Li & Hoiem, 2017), in terms of utilizing new data from novel categories to prevent forgetting of base categories.

Incremental Few-shot Learning: The objective of incremental few-shot learning is to obtain an ability to classify novel classes while preserving the capability to classify base classes, when only a few labeled samples are given for novel classes. In general, the backbone network and classification weights for base classes are pretrained by regular supervised learning and are fixed afterwards. To our best knowledge, prior works on incremental few-shot learning focus on generating the classification weights for novel classes. The Weights Imprinting method of (Qi et al., 2018) computes the prototypes of novel classes and use them as classification weights for novel classes. Another method, Learning without Forgetting of (Gidaris & Komodakis, 2018) learns to generate the novel classification weights by a meta-learned weight generator that takes novel prototypes and base weights as inputs. LwoF utilizes an attention-based mechanism to exploit the relation between base classes and novel classes in the generation of novel weights. Yet another prior work, Attention Attraction Network of (Ren et al., 2019), trains the novel classification weights by a gradient-based optimization process using cross entropy loss from a few labeled samples of the novel classes until they converge. Since the loss for training novel weights is computed only with the samples of novel classes, the catastrophic forgetting issue for base classes may arise. To prevent this, the attention-based regularization method is applied. The regularization loss is provided by a meta-learned attention attractor network. The attention attractor network generates attractor vectors using the base classification weights, and regularization loss is computed based on the Mahalanobis distances between the novel classification weights and attractor vectors. These prior works on incremental few-shot learning focus only on learning to construct novel classifiers. In contrast, our XtarNet aims to extract novel representation which cannot be provided by the pretrained backbone alone.

4. Experimental Result

We follow the settings of incremental few-shot classification tasks proposed in (Gidaris & Komodakis, 2018; Ren et al., 2019) which are based on the *mini*ImageNet and *tiered*ImageNet datasets. Pretraining is done with $\mathcal{D}_{\text{base}}$ consisting of 64 and 200 base categories for *mini*ImageNet and *tiered*ImageNet experiments, respectively. In the meta-training phase, 5 novel categories are selected from $\mathcal{D}_{\text{novel/train}}$ for each episode. For *tiered*ImageNet experiments, $\mathcal{D}_{\text{novel/train}}$ contains additional 151 categories. Following the setting of (Gidaris & Komodakis, 2018; Ren et al., 2019), for *mini*ImageNet experiments, $\mathcal{D}_{\text{base/train}}$ is reused as $\mathcal{D}_{\text{novel/train}}$; 5 base categories are selected as fake novel categories for each episode. For deployment, the validation and test sets of each dataset are used as $\mathcal{D}_{\text{novel/val}}$ and $\mathcal{D}_{\text{novel/test}}$, respectively.

For each episode, the support set is constructed by choosing 1 or 5 examples (or shots) from 5 randomly selected novel categories. The query set consists of samples from 5 novel categories and 5 randomly chosen base catagories. For the meta-training phase, we optimize the number of per-class queries for base and novel categories.

For *mini*ImageNet experiments, we use ResNet12 adopted in (Mishra et al., 2017) as the backbone architecture. MetaCNN consists of a single residual block with the same size as the last residual block of the backbone. Also, MetaCNN takes the output of the third residual block of the backbone as its input. MergeNet contains two separate 4-layer fully-connected networks. For TconNet, two separate 3-layer fully-connected networks with a skip connection for each layer are used to condition the base classifier. Finally, an additional 3-layer fully-connected network with a skip connection for each layer is used to condition the novel classifier. In the meta-learning phase, the MomentumSGD optimizer with an initial learning rate of 0.1 is utilized. The learning rate is dropped by a factor of 10 at every step of 4,000 episodes. For regularization in meta-learning, $l2$ regularization with the ratio 5e-4 is used.

For *tiered*ImageNet, we use the standard ResNet18 architecture. MetaCNN contains two residual blocks with the same size as the last two blocks of the backbone. Other modules are based on the same architectures that are used in

*Table 1. mini*ImageNet 64+5-way results

Methods	1-shot Accuracy	1-shot Δ	5-shot Accuracy	5-shot Δ
Imprint (Qi et al., 2018)	41.34 ± 0.54%	-23.79%	46.34 ± 0.54%	-25.25%
LwoF (Gidaris & Komodakis, 2018)	49.65 ± 0.64%	-14.47%	59.66 ± 0.55%	-12.35%
TapNet (Yoon et al., 2019)	54.38 ± 0.59%	-12.88%	64.02 ± 0.51%	-10.98%
Attention Attractor Networks* (Ren et al., 2019)	54.95 ± 0.30%	-11.84%	63.04 ± 0.30%	-10.66%
XtarNet (Ours)	**54.96** ± **0.61**%	-13.36%	**64.88** ± **0.47**%	-10.41%
Proposed method with Imprint	47.96 ± 0.59%	-12.72%	56.66 ± 0.51%	-11.23%
Proposed method with LwoF	49.71 ± 0.61%	-13.54%	60.13 ± 0.50%	-11.40%

*The Attention Attractor Networks results reflect the reported numbers of Ren et al. All others results are reproduced.

*Table 2. tiered*ImageNet 200+5-way results

Methods	1-shot Accuracy	1-shot Δ	5-shot Accuracy	5-shot Δ
Imprint (Qi et al., 2018)	40.83 ± 0.45%	-22.29%	53.87 ± 0.48%	-17.18%
LwoF (Gidaris & Komodakis, 2018)	53.42 ± 0.56%	-9.59%	63.22 ± 0.52%	-7.27%
TapNet (Yoon et al., 2019)	55.42 ± 0.59%	-8.09%	65.38 ± 0.53%	-6.37%
Attention Attractor Networks* (Ren et al., 2019)	56.11 ± 0.33%	-6.11%	65.52 ± 0.31%	-4.48%
XtarNet (Ours)	**56.57** ± **0.60**%	-8.02%	**66.52** ± **0.49**%	-6.31%
Proposed method with Imprint	56.37 ± 0.56%	-8.89%	65.56 ± 0.50%	-6.82%
Proposed method with LwoF	55.41 ± 0.57%	-9.33%	65.68 ± 0.51%	-6.68%

*The Attention Attractor Networks results reflect the reported numbers of Ren et al. All others results are reproduced.

the *mini*ImageNet experiments. In the meta-learning phase, the MomentumSGD optimizer with an initial learning rate of 0.1 is utilized. The learning rate is dropped by a factor of 10 at every step of 4,000 episodes. For regularization in meta-learning, $l2$ regularization with the ratio 3e-3 is used.

4.1. Evaluation Metrics

We can evaluate the proposed model using joint accuracy, which is evaluated using query sets sampled jointly from base and novel classes. The joint accuracy indicates the capability of models to handle base and novel queries together. Also, the gaps between joint accuracy and individual accuracies for base and novel classes are evaluated. Individual accuracy for base (or novel) classes are obtained by classifying the $\mathcal{Q}_{\text{base}}$ (or $\mathcal{Q}_{\text{novel}}$) query set using only the base (or novel) classifier. The gaps for base and novel classes are denoted by Δ_a and Δ_b, respectively. The average of these gaps $\Delta = \frac{1}{2}(\Delta_a + \Delta_b)$ is evaluated.

4.2. Similarity Metrics

For XtarNet, we use the Euclidean distance metric in classification. For the pretraining, the backbone network is also trained with the Euclidean distance metric. We evaluate other two versions of our proposed method with relatively simple prior methods of Imprint and LwoF, respectively. We use cosine similarity metric for both methods with Imprint and LwoF. For these cases, a backbone is also pretrained with the cosine similarity metric.

4.3. Results

In Tables 1 and 2, we present the results of incremental few-shot learning experiments with the *mini*ImageNet and *tiered*ImageNet datasets. All results for prior methods except for Attention Attractor Networks are reproduced here. The Attention Attractor Networks results shown in the tables are the published numbers of (Ren et al., 2019). We were not able to reproduce similar results for Attention Attractor Networks while we could do so for all other methods. The use of TapNet of (Yoon et al., 2019) for incremental few-shot learning is straightforward and gives competitive performance as shown. For our XtarNet, using TapNet for creating initial novel classifier was a natural choice as it provides solid base performance while being simple. Its projection space is also used in running XtarNet. The performance is evaluated by the averaged accuracy with a 95% confidence interval over 600 test episodes. The Δ values are also evaluated. For each test episode, five base and five novel classes are randomly chosen. From the selected classes, 15 per-class test queries are considered.

For 5-shot *mini*ImageNet case, XtarNet shows the best joint accuracy and Δ value. For 1 and 5-shot *tiered*ImageNet cases which are more complicated than *mini*ImageNet cases,

Table 3. Ablation studies for *tiered*ImageNet

Methods	**5-shot** Accuracy ↑	Δ ↓
Imprint (Qi et al., 2018)	53.87 ± 0.48%	-17.18%
MetaCNN with Imprint	63.15 ± 0.49%	-7.32%
MetaCNN + MergeNet with Imprint	64.62 ± 0.51%	-7.09%
MetaCNN + MergeNet + TconNet with Imprint	65.56 ± 0.50%	-6.82%

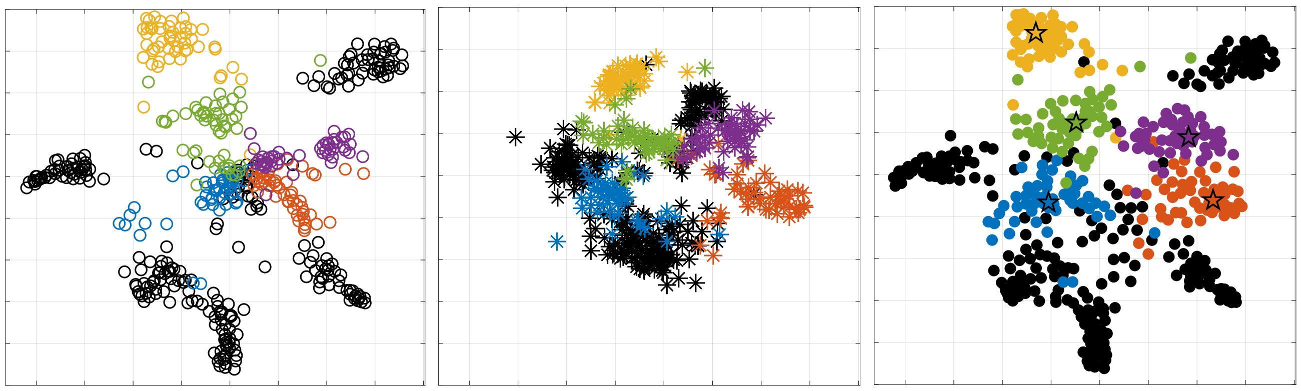

(a) Features from the pretrained backbone (b) Features from MetaCNN (c) Combined features

Figure 2. tSNE plots of queries from a test episode for the proposed TAR method built on Imprint

XtarNet achieves the best joint accuracies with considerable margins. For Δ values, XtarNet shows the second best values for 1-shot *mini*ImageNet case and 1 and 5-shot *tiered*ImageNet cases.

Moreover, the proposed methods in conjunction with Imprint and LwoF also yield significant performance advantages over the base Imprint and LwoF methods. In particular, for Imprint which shows severely degraded performance for incremental few-shot learning tasks, remarkable gains are shown all test cases. For 1 and 5-shot *tiered*ImageNet, the proposed idea combined with Imprint shows even better joint accuracies than existing incremental few-shot learning algorithms.

4.4. Ablation Studies

Table 3 shows evaluation results of one-by-one employment of our meta-trained modules MetaCNN, MergeNet and TconNet. Our ablation studies are done with the Imprint method which does not contain any meta-trained modules for preparing novel classifier weights, so that the results highlight the gains obtained by applying the concept of TAR. Adding just MetaCNN yields a large gain over the baseline Imprint method. Adding MergeNet gives another 1.5% or so gain; adding the TconNet module finally yields yet another 0.9% gain, brining the combined gain of more than 11% overall.

5. Analysis of Task-Adaptive Representation

5.1. tSNE Analysis on Base and Novel Features

In Fig. 2, tSNE analysis is presented for illustrating how MetaCNN extracts novel features and how MergeNet combines them with base features to acquire effective combined features. The proposed method with Imprint is considered for emphasizing gains obtained by employing the concept of TAR. We consider 5-shot *tiered*ImageNet incremental few-shot classification tests. For the analysis, features of 50 per-class queries from a test episode are considered. In the three subfigures of Fig. 2, the black-colored symbols present base class queries from five selected base categories, and the five colored symbols indicate queries from five novel categories. In Fig. 2a, weighted base feature vectors $\boldsymbol{\omega}_{\text{pre}} \odot f_\theta(\mathbf{x})$ of queries are presented with a circle symbol $\circ$. The queries from base classes, the black symbols, are clustered and show good separation. The good clustering and separation make sense because the backbone is pretrained to classify base classes. However, the weighted base feature vectors of queries from novel classes do not show good clustering behavior. In particular, the four novel classes depicted with green, blue, purple and orange colors are severely stuck together and indistinguishable. In Fig. 2b, the weighted novel feature vectors of queries from MetaCNN are illustrated with a star symbol $*$. In comparison with the base feature vectors, MetaCNN clearly provides improved features for identifying novel classes. Finally, the combined feature vectors are presented with a bullet symbol • in Fig. 2c. For base class queries, they remain well-clustered as in Fig 2a. On the other hand, the combine feature vectors of novel classes have changed dramatically. By adopting the features from MetaCNN, the combined features are reasonably clustered in comparison with the base features depicted in Fig. 2a. The per-class averages of the combined feature vectors are presented with the symbols ☆ which work as classification weights of novel classes for the proposed method with Imprint. The per-class averages of novel classes become more separated so that they can provide an appropriate novel classifier. This is the reason why adopting our methods show considerable improvement over the baseline Imprint method.

5.2. Quality of Clustering

Let us take a closer look at how the proposed TAR idea improves upon Imprint. To measure the quality of clustering for each category, the sum of squared distances or errors (SSE) between embedded queries and their centroid is evaluated. The SSE is calculated as follows:

$$\mathrm{SSE}_k = \sum_{\mathbf{x}\in\mathcal{Q}_k} \left\|\mathbf{c}_k^* - \boldsymbol{\omega}_{\text{pre}} \odot f_\theta(\mathbf{x}) + \boldsymbol{\omega}_{\text{meta}} \odot g(a_\theta(\mathbf{x}))\right\|^2 \quad (7)$$

where $\mathcal{Q}_k$ is a set of query samples from class k. Meanwhile, the SSE of the Imprint method is calculated as:

$$\mathrm{SSE}_k = \sum_{\mathbf{x}\in\mathcal{Q}_k} \left\|\mathbf{c}_k - f_\theta(\mathbf{x}))\right\|^2. \quad (8)$$

For each test episode for 5-shot 200+5 *tiered*ImageNet classification, the SSE values for each method are averaged for base and novel categories. 50 queries are considered for each class. Results are averaged over 600 episodes. Table 4 shows the averaged SSE values of Imprint as well as the combined method. Let us focus on the first column for each base and novel case in Table 4. It is clearly shown that the concept of TAR significantly improves the quality of clustering for both base and novel categories. From the SSE analysis, it is apparent that the TAR enhances the quality of clustering for both base and novel categories.

To consider the interference from adjacent clusters, SSE values can be normalized by the squared distance to the nearest interfering centroid (denoted by nSSE). The results are reported in the right column of each base and novel case in Table 4. When the inference is considered, the TAR shows much stronger effect on novel classes.

5.3. Entropy

Let us now consider entropy analysis with two information-theoretic metrics: cross entropy $\mathcal{E}$ and Shannon entropy $\mathcal{H}$. We use the same setting as in the analysis for the quality

Table 4. Clustering quality analysis for *tiered*ImageNet results

	Base		Novel	
Methods	SSE	nSSE	SSE	nSSE
Imprint	10.65	81.02	3.96	250.31
Combined	4.94	76.92	2.38	179.51
Reduction Ratio	-53.7%	-5.06%	-40.0%	-28.3%

of clustering. For each test episode from *tiered*ImageNet, 50 per-class queries are selected and the averages of their entropy values are evaluated. For two groups of base and novel categories, the means of the averaged entropy values are calculated. A smaller cross-entropy value means that inference on labels is more accurate. A smaller Shannon entropy value indicates that inference is more confident. The results in Table 5 shows that inference on test queries becomes more accurate and confident by means of the TAR.

Table 5. Entropy analysis for *tiered*ImageNet results

	Base		Novel	
Methods	$\mathcal{E}$	$\mathcal{H}$	$\mathcal{E}$	$\mathcal{H}$
Imprint	4.78	5.31	4.97	5.31
Combined	1.61	1.48	0.92	1.37
Reduction Ratio	-66.3%	-72.1%	-81.5%	-74.2%

6. Conclusions

We proposed an incremental few-shot learning algorithm which learns to extract task-adaptive representation for classifying both base and novel categories. Built on a pretrained model, our proposed method XtarNet employs three essential modules that are meta-trained. For a given new task, XtarNet extracts novel features which cannot be captured by the pretrained backbone. The novel feature is then combined with the base feature captured by the backbone. The mixture of these features constructs the TAR, facilitating incremental few-shot learning. The base and novel classifier quickly adapt to the given task by utilizing the TAR. The concept of the task-adaptive representation can be used in conjunction with known incremental few-shot learning methods, and significant performance gains are achieved. For the incremental few-shot learning benchmarks using *mini*ImageNet and *tiered*ImageNet, our XtarNet achieves state-of-the-art accuracies.

References

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135, 2017.

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

Kemker, R. and Kanan, C. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SJ1Xmf-Rb`.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *NIPS 2017 Workshop on Meta-Learning*, 2017.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkQqq0gRb`.

Oreshkin, B. N., Lacoste, A., and Rodriguez, P. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, 2018.

Perez, E., Strub, F., Vries, d. H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Qi, H., Brown, M., and Lowe, D. G. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5822–5830, 2018.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Ren, M., Liao, R., Fetaya, E., and Zemel, R. Incremental few-shot learning with attention attractor networks. In *Advances in Neural Information Processing Systems*, pp. 5276–5286, 2019.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4080–4090, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Yoon, S. W., Seo, J., and Moon, J. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. *arXiv preprint arXiv:1905.06549*, 2019.

SUPPLEMENTAL MATERIAL
XtarNet: Learning to Extract Task-Adaptive Representation for Incremental Few-Shot Learning

Sung Whan Yoon [1] **Do-Yeon Kim** [2] **Jun Seo** [2] **Jaekyun Moon** [2]

1. Details of Backbone Networks

For *mini*ImageNet experiments, we utilized ResNet12, a residual network with 12 layers, as architecture of the backbone network. ResNet12 consists of four residual blocks having 64, 96, 128 and 256 respective channels. Each residual block contains three 3×3 convolutional layers followed by a batch normalization and an ReLU activation. The residual connection of each block consists of a single 3×3 convolutional layer and a batch normalization layer. The output of the residual connection is delivered to the input of the last activation layer of the residual block. At the end of each residual block, 2×2 max-pooling is applied. To reduce the dimension of the feature vector, a global average pooling is applied to the output of the last residual block.

For *tiered*ImageNet experiments, we employed the standard ResNet18, which consists of 18 convolutional layers, as the backbone network. ResNet18 is well-known, and we forgo a detailed description here.

2. Details of Meta-Trained Modules

The details of MetaCNN module are described in the main paper. We utilize fully-connected networks for MergeNet and TconNet. For the *mini*ImageNet experiments, MergeNet consists of two separate 4-layer fully-connected networks. These two component networks have the same architecture. The size of the input layer of each 4-layer fully-connected network is 512, which is the length of a concatenated feature vector. For the hidden layers and the output layer, the size of each layer is 256. For each of first three fully-connected layers, an ReLu activation is employed. For the last fully-connected layer, a sigmoid layer is employed. TconNet consists of three 3-layer fully-connected networks. Two of them are for computing the conditioning vector for the base classifier. These two fully-connected networks have a skip connection at each layer. The size of each fully-connected layer is 256, which is the length of a combined feature vector. ReLu activation is used at every layer. At each output of these two component fully-connected networks, we multiply the output vector by the constant 0.1 for scaling. The remaining 3-layer fully-connected network is for conditioning of the novel classifier. The size of each fully-connected layer is 64, which is the number of the base categories. The last two fully-connected layers have skip connections. ReLu activation is used at every layer.

For the *tiered*ImageNet experiments, the architectures of MergeNet and TconNet are the same as that we used in the *mini*ImageNet experiments, except for the size of the fully-connected layers. Because the length of the feature vector becomes double, the size of every fully-connected layer of MergeNet and TconNet also doubles.

3. Hyperparameter Settings

Tables 1 and 2 show the hyperparameter settings for the *mini*ImageNet and *tiered*ImageNet experiments. The number of per-class queries in each training episode, steps of the learning rate decay and $l2$ weight decay rates are shown.

The settings in Table 1 are used in both 1-shot and 5-shot *mini*ImageNet experiments. We use 15 per-class queries in each training episode for the proposed TAR method used in conjunction with Imprint and LwoF, respectively. On the other hand, for XtarNet, the TAR method built on TapNet, the number of per-class queries is changed from 5 to 15 for base categories, and from 25 to 15 for novel categories at 2,000 episodes. For all cases, we use the Momentum SGD optimizer with an initial learning rate of 0.1. The learning rate decays by a factor of 10 at every step of 4,000 episodes. $l2$ weight decay is set to be 5e-4 for all *mini*ImageNet experiments. Also, a and b, the hyperparameters of TconNet, are both set to 0.1.

The settings for *tiered*ImageNet are shown in Table 2. The 1-shot and 5-shot experiments are done with the same settings. For all cases, identical hyperparameters are used as shown

[1]School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan, Korea [2]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. Correspondence to: Sung Whan Yoon <shyoon8@unist.ac.kr>.

Work in Progress.

in Table 2.

4. Study on Mixture Weight Vectors

4.1. Magnitude Analysis

Fig. 1 shows the magnitude of the mixture weight vectors $\boldsymbol{\omega}_{\text{pre}}$ and $\boldsymbol{\omega}_{\text{meta}}$ during the meta-training phase. We consider the 5-shot 200+5 *tiered*ImageNet case. The mixture weight vector for the feature from the pretrained backbone is seen to have a larger magnitude than the weight vector for the feature from the MetaCNN. The magnitude of $\boldsymbol{\omega}_{\text{pre}}$ fluctuates and slightly decreases until the learning rate decay at 4,000 episodes. Afterwards, the magnitude gradually increases and eventually saturates. For the mixture weight vector $\boldsymbol{\omega}_{\text{meta}}$ from the MetaCNN, the magnitude steadily increases over time and also saturates soon. An obvious observation is that the magnitudes of the mixture weight vectors become quite stable as the meta-training phase is over. Also, given the difference in the magnitudes between the two weight vectors, we conjecture that XtarNet more actively utilizes the feature from the pretrained backbone, while also significantly leveraging the feature from MetaCNN.

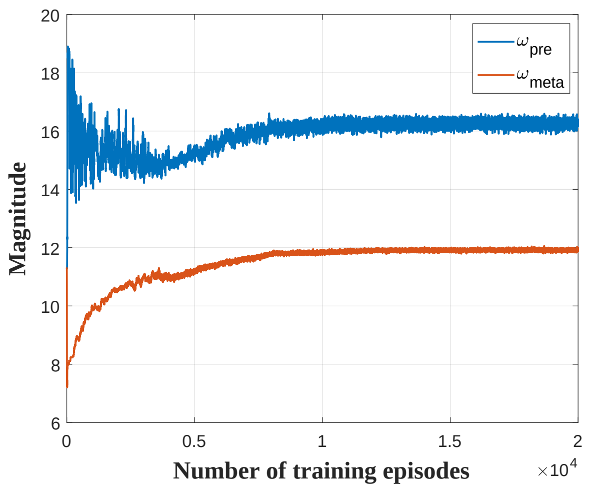


Figure 1. Magnitude of mixture weight vectors during the meta-training phase

4.2. Principal Component Analysis of Mixture Weight Vectors

We analyze the mixture weight vectors in the combined feature space constructed by query samples. The 5-shot 200+5 *tiered*ImageNet experiment is also considered. First, we did a principal component analysis (PCA) of 500 query samples for a test episode. The magnitude of the projected mixture weight vector to each principal component is computed in order to figure out how much power of mixture weight vector aligns with each principal component. In Fig. 2, it is shown that the magnitudes of mixture weight vectors for a backbone and MetaCNN are focused mainly on the first 150 or so principal axes of query samples. Consequently, the mixture weight vectors aim much of their attention at the critical dimensions where query samples show larger variances.

5. Dataset Statistics

In this section, we present the detailed statistics of the *mini*ImageNet and *tiered*ImageNet datasets that we use in the incremental few-shot learning tests. We follow the data splits used in prior works of (?) and (?). The data splits of *mini*ImageNet are presented in Table 3. For the *mini*ImageNet experiments, the training set $\mathcal{D}_{\text{base/train}}$ (denoted by 'Train-Train'), which is for pretraining of base classes, is reused as the training set $\mathcal{D}_{\text{novel/train}}$ (denoted by 'Train-Val') for novel classes. Since the data split $\mathcal{D}_{novel/train}$ contains only base classes, some randomly selected base classes are used as fake novel classes. This process is suggested in the prior work (?) and (?). As a result, the model is meta-trained with 59+5-way incremental few-shot learning episodes instead of 64+5-way episodes. However, the final model is evaluated with 64+5-way incremental few-shot learning episodes.

On the other hand, for the *tiered*ImageNet experiments, the $\mathcal{D}_{\text{base/train}}$ and $\mathcal{D}_{\text{novel/train}}$ are different. The original training split of *tiered*ImageNet which contains 351 classes is divided into two splits of 'Train-A' and 'Train-B' containing disjoint classes. The 'Train-A' is then further divided into training/validation/test sets of 'Train-A-Train','Train-A-Val' and 'Train-A-Test' with disjoint samples. These splits are for base classes. For novel categories, 151 extra classes of a set 'Train-B' are used for novel classes in the meta-training phase. For validation and test, sets denoted by 'Val' and 'Test' are used respectively. These splits for novel categories contain disjoint classes.

6. Few-shot Classification Accuracies

We also evaluate few-shot classification accuracies of XtarNet, which is trained as an incremental few-shot learner supposed to handle both base and novel categories. The accuracy rates on *mini*ImageNet and *tiered*ImageNet datasets, as shown in Tables 5 and 6, are all from previously published results, except for our XtarNet. XtarNet yields performance that is only slightly worse than the existing ResNet12-based methods focusing on few-shot learning tasks in the upper portions of the tables.

It can be seen that the known incremental few-shot learning methods, LwoF and Attention Attractor Networks, show significantly degraded performance compared to the ResNet12-based learners specialized to few-shot learning (few-shot

Table 1. Hyperparameter settings for 64+5 *mini*ImageNet experiments

Methods	Optimizer	N_q base	N_q novel	lr decay step	$l2$ decay rate
XtarNet	mSGD	5→15*	25→15*	4,000	5e-4
Proposed TAR with Imprint	mSGD	15	15	4,000	5e-4
Proposed TAR with LwoF	mSGD	15	15	4,000	5e-4

*The number of queries for base and novel categories are changed at 2,000 episode.

Table 2. Hyperparameter settings for 200+5 *tiered*ImageNet experiments

Methods	Optimizer	N_q base	N_q novel	lr decay step	$l2$ decay rate
XtarNet	mSGD	5	25	4,000	3e-3
Proposed TAR with Imprint	mSGD	5	25	4,000	3e-3
Proposed TAR with LwoF	mSGD	5	25	4,000	3e-3

classification performance of Attention Attractor Networks for *tiered*ImageNet is not available). Compared to these prior incremental few-shot learners, our XtarNet shows much better accuracies, indicating relatively strong few-shot learning performance although its design aimed at incremental few-shot learning tasks.

7. Time Complexity

In Tables 1 and 2 of the main paper, XtarNet shows state-of-the-art accuracies on incremental few-shot learning tasks. One may argue, however, that the performance advantage over the reported results of Attention Attractor Networks is small, especially for 1-shot experiments. While this may be true, we bring up an important point here. Our XtarNet has significant advantage in time complexity over Attention Attractor Networks. The meta-trained modules in XtarNet operate without any recursive optimization procedures, whereas Attention Attractor Networks require several tens of inner-loop optimization steps for obtaining novel classifier weights (). See Steps 6 through 9 in Algorithm 1 of (). This recursive process in Attention Attractor Networks causes a substantial latency relative to our XtarNet during query processing in both training and inference. Furthermore, during training, Attention Attractor Networks actually employ double inner-loop optimization based on Recurrent Back-Propagation of () to compute gradients. Steps 15 through 17 of Algorithm 1 of () represent the second inner loop. This further slows down meta-training of the learner relative to our method.

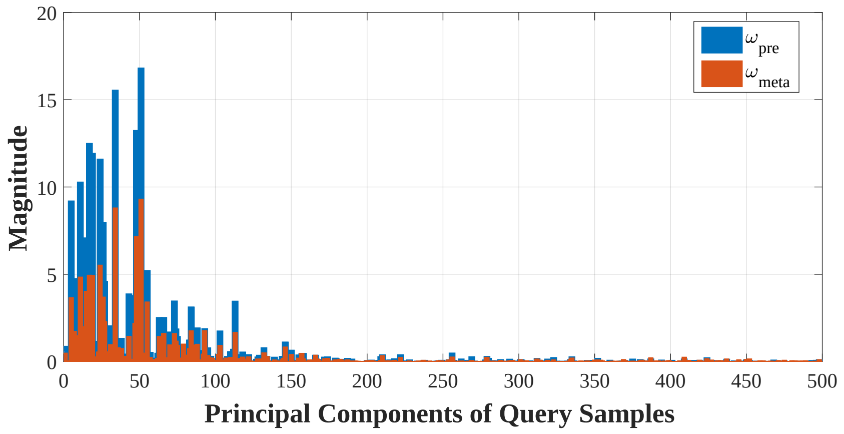


Figure 2. Principal Component Analysis for Mixture Weight Vectors

Table 3. Dataset statistics of *mini*ImageNet

Classes	Purpose	Split	N_{classes}	N_{samples}
Base ($\mathcal{D}_{\text{base}}$)	Train	Train-Train ($\mathcal{D}_{\text{base/train}}$)	64	38,400
	Val	Train-Val ($\mathcal{D}_{\text{base/val}}$)	64	18,748
	Test	Train-Test ($\mathcal{D}_{\text{base/test}}$)	64	19,200
Novel ($\mathcal{D}_{\text{novel}}$)	Train	Train-Train ($\mathcal{D}_{\text{novel/train}} \leftarrow \mathcal{D}_{\text{base/train}}$)	64	38,400
	Val	Val ($\mathcal{D}_{\text{novel/val}}$)	16	9,600
	Test	Test ($\mathcal{D}_{\text{novel/test}}$)	20	12,000

Table 4. Dataset statistics of *tiered*ImageNet

	Purpose	Split	N_{classes}	N_{samples}
Base ($\mathcal{D}_{\text{base}}$)	Train	Train-A-Train ($\mathcal{D}_{\text{base/train}}$)	200	203,751
	Val	Train-A-Val ($\mathcal{D}_{\text{base/val}}$)	200	25,460
	Test	Train-A-Test ($\mathcal{D}_{\text{base/test}}$)	200	25,488
Novel ($\mathcal{D}_{\text{novel}}$)	Train	Train-B ($\mathcal{D}_{\text{novel/train}}$)	151	193,996
	Val	Val ($\mathcal{D}_{\text{novel/val}}$)	97	124,261
	Test	Test ($\mathcal{D}_{\text{novel/test}}$)	160	206,209

Table 5. Few-shot classification accuracies for 5-way *mini*ImageNet

		5-way *mini*ImageNet	
Methods	backbone	1-shot	5-shot
MAML	Conv4	48.70 ± 1.84%	63.15 ± 0.91%
Prototypical Nets	Conv4	49.42 ± 0.78%	68.20 ± 0.66%
Relation Nets	Conv4	50.44 ± 0.82%	65.32 ± 0.70%
Transductive Propagation Nets	Conv4	55.51 ± 0.86%	69.86 ± 0.65%
TADAM	ResNet12	58.5 ± 0.3%	76.7 ± 0.3%
TapNet	ResNet12	61.65 ± 0.15%	76.36 ± 0.10%
MetaOpt	ResNet12	62.64 ± 0.62%	78.63 ± 0.46%
Cross Attention Nets*	ResNet12	63.85 ± 0.48%	79.44 ± 0.34%
LwoF	ResNet12	55.45 ± 0.89%	70.92 ± 0.35%
Attention Attractor Networks	ResNet12	55.75 ± 0.51%	70.14 ± 0.44%
XtarNet (Ours)	ResNet12	60.03 ± 0.30%	75.03 ± 0.30%

*The CAN version with high-confidence query samples yields higher accuracy but is not shown here for fairness reasons.

Table 6. Few-shot classification accuracies for 5-way *tiered*ImageNet

		5-way *tiered*ImageNet	
Methods	backbone	1-shot	5-shot
MAML	Conv4	51.67 ± 1.81%	70.30 ± 1.75%
Prototypical Nets	Conv4	53.31 ± 0.89%	72.69 ± 0.74%
Relation Nets	Conv4	54.48 ± 0.93%	71.31 ± 0.78%
Transductive Propagation Nets	Conv4	59.91 ± 0.94%	73.30 ± 0.75%
TapNet	ResNet12	63.08 ± 0.15%	80.26 ± 0.12%
MetaOpt	ResNet12	65.99 ± 0.72%	81.56 ± 0.53%
Cross Attention Nets*	ResNet12	69.89 ± 0.51%	84.23 ± 0.37%
LwoF	ResNet12	59.79 ± 0.68%	75.77 ± 0.54%
XtarNet (Ours)	ResNet12	63.08 ± 0.30%	79.20 ± 0.30%

*The CAN version with high-confidence query samples yields higher accuracy but is not shown here for fairness reasons.