SEEN: Few-Shot Classification with SElf-ENsemble

Weisen Jiang*†, Yu Zhang* and James T. Kwok†
*Department of Computer Science and Engineering, Southern University of Science and Technology
†Department of Computer Science and Engineering, Hong Kong University of Science and Technology
wjiangar@cse.ust.hk, yu.zhang.ust@gmail.com, jamesk@cse.ust.hk

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9533845

***Abstract*—Few-shot classification aims at learning new concepts with only a few labeled examples. In this paper, we focus on metric-based methods that have achieved state-of-the-art performance. However, they classify query examples based on embeddings extracted from only the last layer. These embeddings tend to be class-specific and may not generalize well to novel classes or domains. To alleviate this problem, we propose the SElf-ENsemble (SEEN) that leverages embeddings from multiple layers. Specifically, a base classifier is built for each of the last few layers, and the resultant base classifiers are then combined together. Experiments on various benchmark datasets demonstrate that the proposed SEEN method outperforms existing methods in both standard few-shot classification and cross-domain few-shot classification scenarios.**

***Index Terms*—few-shot learning, meta-learning**

I. Introduction

Deep neural networks have achieved great success in many visual recognition tasks [1], [2]. Training these networks usually requires massive labeled data, which limits their applications since collecting and labeling massive data is expensive or even impossible. For example, campaign budgets restrict researchers to annotate millions of examples and rare animal species are difficult to capture. Training a deep network from scratch on a small number of examples easily leads to overfitting and poor generalization. On the other hand, humans are capable of learning new concepts with little supervision.

Recently, *few-shot classification* [3]–[5] has made significant progress in bridging this gap. Many approaches have been proposed, and they consist of two core components: a *feature extractor* which maps the input space to a feature space, and a *classifier* which maps the feature space to the label space. While the feature extractor can be enhanced by techniques such as increasing the network depth, most prior studies focus on enhancing the classifier. For example, ProtoNet [6] classifies examples based on the Euclidean distance between query examples and prototypes, RelationNet [7] learns a deep distance metric, MatchingNet [8] incorporates the classifier with an attention mechanism, and GNN [9] employs graph convolution blocks as the metric function. A limitation with existing few-shot algorithms is that the classification output is trained using embeddings extracted from the last layer only but ignoring other layers. Features extracted from the last layer may be too task-specific to generalize to unseen classes or novel domains. As an illustration, Fig. 1 compares the few-shot classification accuracies of three prototype classifiers [10],

Corresponding author: Yu Zhang

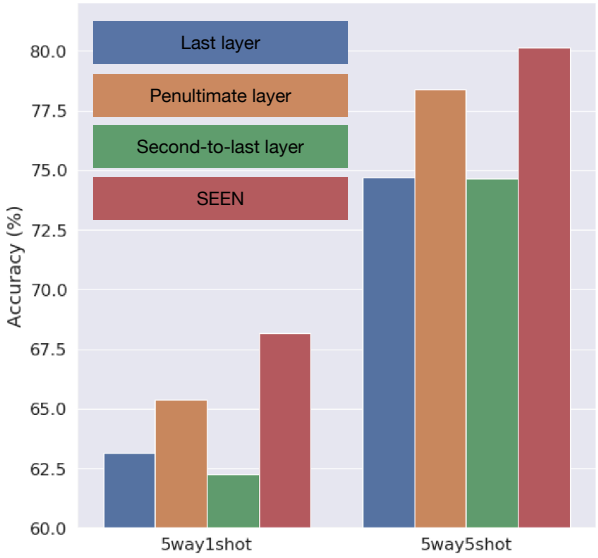


Fig. 1. Performance of 3 prototype classifiers and their ensemble on the meta-testing set of *mini*-ImageNet using ResNet-12. Best viewed in color.

each of which is built on one of the last three layers of a pretrained residual network (ResNet-12). As can be seen, the classifier trained from the penultimate layer outperforms those trained from the last and second-to-last layers. Moreover, learners based on a single layer are usually weak. As shown in Fig. 1, in the 1-shot setting, the three base learners have classification accuracies of around 65%, which are much lower than those in the 5-shot setting (approximately 78%).

Inspired by ensemble learning, in this paper we propose a SElf-ENsemble (SEEN) to leverage the last few layers of the neural network. Specifically, we first construct a classifier for each of these layers, and then combine their predictions to make a final prediction. We study three combination strategies: (i) uniform averaging, which treats individual classifiers equally important, (ii) weighted averaging, and (iii) stacking. Experimental results show that the proposed SEEN method achieves state-of-the-art few-shot classification performance on benchmark datasets, including *mini*-ImageNet, *tiered*-ImageNet, and CIFAR-FS. Moreover, SEEN (with uniform averaging) also outperforms existing works on cross-domain few-shot classification for the *mini*-ImageNet $\rightarrow$ CUB task.

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

II. Related Work

Few-shot classification. Existing studies on few-shot classification [3]–[5] aim at learning from only a few labeled examples. A few-shot model is learned from a meta-training set, with the help of a meta-validation set for model selection, and evaluated on a meta-testing set. Each of those meta-sets contains a set of tasks, and their label sets are disjoint. In n-way k-shot classification, each task is given a support set with k examples for each of the n classes for training, and a query set containing unseen examples for evaluation.

Recent few-shot classification models can be categorized as an *optimization-based* or *metric-based* method. Optimization-based methods allow rapid model adaptation with limited examples using gradient-based algorithms. One line of work focuses on learning a good initialization such that a good model can be obtained on a novel task by performing only a few updates. A representative method is the Model-Agnostic Meta-Learning (MAML) [11], which uses gradient descent methods to find an initialization with better generalization ability. To reduce the complexity of computing the expensive second-order derivatives, first-order MAML (FOMAML) [11] ignores the Hessian matrix, while REPTILE [12] approximates it with a combination of gradients. Another line of work focuses on learning a good optimizer. For example, methods in [13], [14] learn the parameter update rule with recurrent neural networks, Meta-SGD [15] learns feature-wise learning rate schedules, while Meta-Curvature [16], T-Nets [17] and WarpGrad [18] learn to capture the curvatures. While optimization-based methods allow rapid adaptation, they suffer in the presence of domain shifts [3]. Moreover, as the initialization/optimizer fixes the network structure and output dimension of the last layer, they fail to generalize to classification tasks with different numbers of classes.

On the other hand, metric-based methods have two core components: (1) a *feature extractor* which maps from the input space to a feature space, and (2) a *classifier* which maps from the feature space to the label space. Several recent works focus on designing task- or class-dependent feature extractors to provide more discriminative and representative features. For example, CAML [19] introduces a class-dependent transformation to capture inter-class dependencies. TADAM [20] inserts task-conditioning layers to the feature extractor. CTM [21] employs a feature mask to utilize inter-class uniqueness and intra-class commonality structures. Other methods focus on enhancing the classifier or designing proper distance metrics. For instance, ProtoNet [6] classifies examples based on the Euclidean distance between the query examples and prototypes. R2-D2 [22] and MetaOptNet [23] adopt linear models as the classifier due to their computational efficiency. MatchingNet [8] employs the negative cosine similarity with an attention mechanism. GNN [9] uses graph convolution blocks as the metric function to do the classification. RelationNet [7] learns a deep metric for classification.

Data augmentation. Data augmentation can also help mitigate the data scarcity problem in few-shot classification. For example, DAGAN [24] trains GAN [25] to transfer image styles, while SGM [26] learns a data generation process for unseen classes. In contrast to augmenting samples, recent studies also focus on task augmentation. CACTUs [27] proposes to generate few-shot tasks from unlabeled data for training, while MAXL [28] trains few-shot learning tasks with auxiliary tasks to improve the generalization ability. All the above augmentation approaches are general strategies such that they can be combined with existing few-shot classification algorithms and so we do not compare with them in the experiments.

III. SElf-ENsemble (SEEN)

In this paper, we focus on the metric-based approach for few-shot classification, which has been shown to have promising performance [3], [29]. The proposed SEEN method is inspired by ensemble learning [30], which combines several base learners to build a powerful learner. The diversity among base learners is crucial to ensemble learning. Prior works on few-shot ensemble learning either train diverse models on the same meta-training set [31] or train models on diverse meta-training sets [29]. These approaches are computationally inefficient and time-consuming by training multiple models or gathering multiple datasets. However, as will be seen in the sequel, the proposed SEEN method can mitigate these shortcomings by reusing a network and building multiple classifiers from its different layers.

The following sections will give details of SEEN, which is first trained on the meta-training set $\mathcal{D}^{tr}$, then does model selection and ensemble learning on the meta-validation set $\mathcal{D}^{vl}$, and finally is evaluated on the meta-testing set $\mathcal{D}^{ts}$.

A. Network Architecture

Recall that there are two components in a metric-based few-shot classifier, namely, feature extractor and classifier. As deep networks have shown excellent performance in image classification [1], [2], we use a L-layer neural network f_θ (parameterized by θ) as the feature extractor. As for the classifier, the cosine classifier is simple and yet has better generalization to novel categories than the linear classifier [3], [32], [33]. For a query example x, the cosine classifier computes the posterior probability that x belongs to class c as

$$\mathbb{P}(y=c|x;\theta,\{w_c\}) = \frac{\exp(\tau\cos(w_c, f_\theta(x)))}{\sum_{c'}\exp(\tau\cos(w_{c'}, f_\theta(x)))}, \quad (1)$$

where $w_c \in \mathbb{R}^d$ contains classifier parameters for class c, d is the dimensionality of the feature extracted by f_θ, $\tau > 0$ is the temperature [20], and $\cos(x, y)$ denotes the cosine similarity.

During meta-training, the feature extractor and cosine classifier are trained jointly in a standard supervised learning manner by minimizing the cross-entropy loss as

$$\mathcal{L}(\theta,\{w_c\}) = \sum_{(S,Q)\in\mathcal{D}^{tr}}\sum_{(x,c^*)\in S\cup Q} -\log\mathbb{P}(y=c^*|x;\theta,\{w_c\}), \quad (2)$$

where (S, Q) denotes a task consisting of a support set S and a query set Q, and c^* is the true label of x.

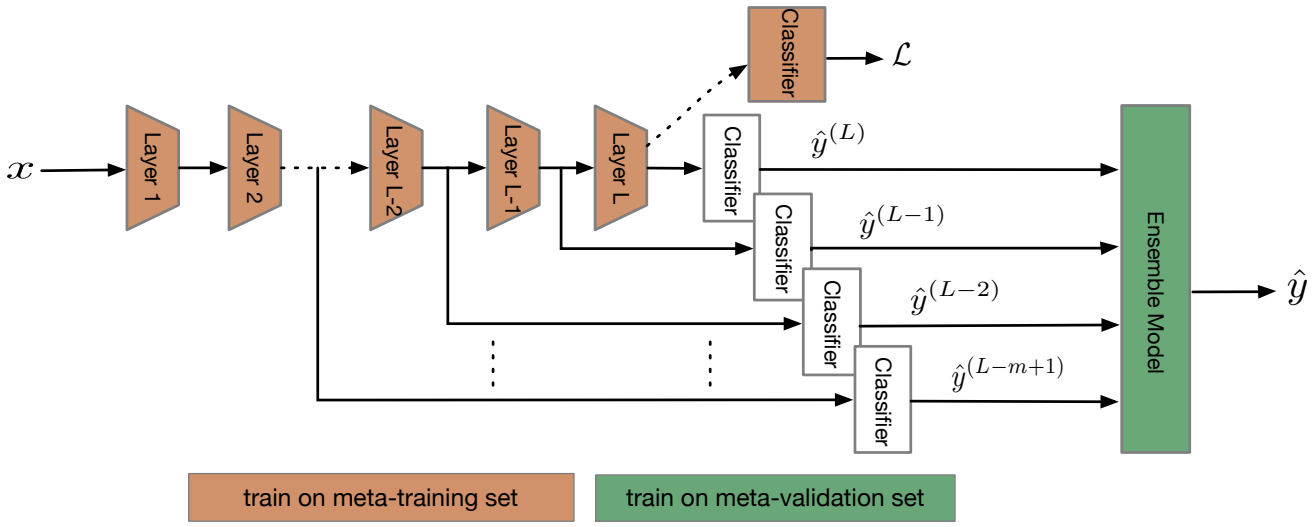


Fig. 2. Overview of the SEEN method. We pretrain a neural network on the meta-training set using standard supervised learning and freeze it as a feature extractor at the meta-validation and meta-testing stages. Unlike prior works that use embeddings extracted from only the last layer, SEEN combines the last m layers. We first build a classifier on each of the last m layers, then combine predictions from all classifiers to make a final prediction. Best viewed in color.

At the meta-testing stage, given an unseen task, retraining the feature extractor on limited support examples may lead to overfitting. To alleviate this problem, a commonly used approach is to freeze the feature extractor and employ a simple classifier on the extracted features.

We consider to replace the cosine classifier by the prototype classifier [6], [10], also called the mean-centroid classifier [29] or the nearest class mean classifier [34]. Let S_c be the subset of examples in the support set S belonging to class c. For $(x, y) \in S_c$, let $f_\theta(x; l)$ be the embedding extracted from the lth layer of the neural network f_θ. The prototype of class c is computed as

$$p_c^{(l)} = \frac{1}{|S_c|} \sum_{(x,y)\in S_c} f_\theta(x; l). \tag{3}$$

Prototype p_c can be viewed as w_c in (1), and be used for the prediction. It is more advantageous than the cosine classifier in that it is parameter-free and thus requires no training, which can help avoid overfitting.

The prototypes obtained from a small set of examples in (3) can have high variance, which impairs performance. To alleviate this problem, recent studies [10], [12], [35] use unlabeled query examples as in transductive learning. For example, BD-CSPN [10] rectifies the prototypes by weighting unlabeled query examples based on their cosine similarities as

$$\bar{p}_c^{(l)} = \sum_{(x,y)\in S_c\cup Q} \mathrm{softmax}(\tau\gamma(x)) f_\theta(x; l), \tag{4}$$

where τ is the temperature in (1), and

$$\gamma(x) = \begin{cases} \cos(p_c^{(l)}, f_\theta(x; l)) & x \in Q \\ 1 & x \in S_c \end{cases}. \tag{5}$$

B. SElf-ENsemble (SEEN)

Existing few-shot classifiers are constructed based on embeddings extracted from the last layer of the neural-network feature extractor but not from the other layers. As demonstrated in Fig. 1, the last layer may be too specific to generalize to novel tasks. To alleviate this problem, we propose to build an ensemble of classifiers, each of which is built on a specific hidden layer. As features extracted from a deep neural network usually transit from general to specific when the layers go deeper [36], this diversity of features from different layers guarantees the diversity of the corresponding classifiers constructed. Although we allow building classifiers for each layer, the first several layers are less useful as they tend to capture local structures such as corners and edges, which are less discriminative for classification [37], [38].

Fig. 2 provides an overview of the proposed SEEN method. After pretraining on the meta-training set, we freeze the feature extractor. At the meta-validation stage, we build m independent base classifiers for each of the last m layers. In the experiments, we use the prototype classifier introduced in Section III-A to evaluate the proposed SEEN. Let $\hat{y}^{(l)}$ be the prediction from the classifier built from the lth layer. Instead of using the prediction from a single layer, SEEN combines their predictions as

$$\hat{y} = \sum_{l=L-m+1}^{L} \phi(x; l)\hat{y}^{(l)}, \tag{6}$$

where $\phi(x; l) \in [0, 1]$ is the weight for the layer-l classifier, and $\sum_{l=L-m+1}^{L} \phi(x; l) = 1$.

The setting of $\phi(x; l)$ is key to the performance. We propose three combination strategies as follows.

1) **Uniform averaging**, which averages predictions from all base classifiers, with $\phi(x; l) = 1/m$ for $l = L -$

$m+1, \ldots, L$. This strategy assumes that all classifiers contribute equally.

2) **Weighted averaging** weights the layer-l classifier as proportional to its performance π_l on the meta-validation set:

$$\phi(x; l) = \frac{\pi_l}{\sum_{l'} \pi_{l'}}. \tag{7}$$

During meta-testing, $\phi(\cdot; l)$'s are frozen and shared across all testing examples.

3) **Stacking** [39] is an example-dependent ensemble method. The intuition is that some examples may be easier to be classified in certain layers. Given a query example, we first construct a context feature from predictions of the m base classifiers:

$$h = \left[\hat{y}^{(L-m+1)}, \ldots, \hat{y}^{(L)}\right] \in \mathbb{R}^{nm}. \tag{8}$$

We then use a meta-learner $\phi_\pi : \mathbb{R}^{nm} \rightarrow [0, 1]^m$, parameterized by π, to generate combination weights for the m classifiers. For example, ϕ_π can be a 2-layer neural network followed by softmax. π is obtained by minimizing the following objective as

$$\mathcal{L}(\pi) = - \sum_{(S,Q)\sim\mathcal{D}^{vl}} \sum_{(x,c^*)\in Q} \log \hat{y}_{c^*}, \tag{9}$$

where $\hat{y}$ is defined in (6). Training the meta-learner may lead to overfitting. Hence, we hold out a subset of the meta-validation set for model selection and early stopping.

Algorithm 1 shows the training procedure of SEEN with stacking. Among the three combination strategies, weighted averaging and stacking require a meta-validation set to learn the combination coefficients.

Algorithm 1 SEEN with stacking.

Require: meta-training set $\mathcal{D}^{tr}$, meta-validation set $\mathcal{D}^{vl}$;
Require: feature extractor f_θ (a L-layer network), meta-learner ϕ_π, stepsize α;
Meta-Training:
Train f_θ on $\mathcal{D}^{tr}$ using Eq. (2), and then freeze f_θ;
Meta-Validation:
while not converged **do**
  Sample a task with support set S and query set Q;
  for layer $l = L - m + 1, \ldots, L$ **do**
    train a classifier on S using layer-l features;
  end for
  For each query example in Q, combine predictions from the m classifiers using Eq. (6);
  Update the meta-learner:

$$\pi \leftarrow \pi + \alpha \nabla_\pi \sum_{(x,c^*)\in Q} \log \hat{y}_{c^*} \tag{10}$$

end while
return f_θ and ϕ_π;

TABLE I
STATISTICS FOR FEW-SHOT CLASSIFICATION DATASETS.

dataset	#classes	#images	size
mini-ImageNet [8]	100	60,000	84×84
tiered-ImageNet [40]	608	779,165	84×84
CIFAR-FS [22]	100	60,000	32×32
CUB [41]	200	11,788	224×224

IV. EXPERIMENTS

In this section, we empirically evaluate the performance of the proposed SEEN method.

A. Setup

Datasets. Four benchmark datasets are used, including the *mini*-ImageNet and *tiered*-ImageNet from ILSVRC-2012 [44], CIFAR-FS from CIFAR-100, and CUB [41]. A summary of those datasets is shown in Table I.

***Mini*-ImageNet** [8] consists of 100 randomly chosen classes from ILSVRC-2012 [44]. Each class contains 600 images of size 84×84. Following [14], the 100 classes are randomly split into 64, 16, 20 classes for meta-training, meta-validation, and meta-testing, respectively.

***Tiered*-ImageNet** [40] is a larger subset of ILSVRC-2012 [44], with 608 classes categorized into 34 high-level categories. We use 20 categories (with 351 classes) for meta-training, 6 categories (with 97 classes) for meta-validation, and 8 categories (with 160 classes) for meta-testing. The category-level division increases the semantic distance between classes from different categories, and makes few-shot classification more challenging.

CIFAR-FS [22] is a recent benchmark for few-shot classification. It has 100 classes from CIFAR-100. Similar to *mini*-Imagenet, the classes are randomly split into 64, 16, and 20 for meta-training, meta-validation, and meta-testing, respectively. Each class has 600 images of size 32×32. The resolution is lower than *mini*-ImageNet, and thus more difficult.

CUB is a dataset of 200 bird species and has 11,788 images. Following [3], we randomly split the classes into 100, 50 and 50 for meta-training, meta-validation, and meta-testing, respectively. We use this dataset in the experiment on cross-domain few-shot classification in Section IV-C.

Backbone Networks. The feature extractor is implemented by two commonly used backbones: (i) a shallow 4-layer convolutional network (Conv-4-64 [8]) and (ii) a deep residual network (ResNet-12 [1]). *Conv-4-64* has 4 repeated convolutional blocks, in which each block consists of a 3×3 convolutional layer with 64 filters, a batch normalization layer, a ReLU nonlinear layer, and a 2×2 max-pooling layer. *ResNet-12* consists of 4 residual blocks, in which each block stacks three 3×3 convolutional layers with batch normalization and ReLU nonlinearity followed by 2×2 max-pooling. The number of filters in each block starts from 64, and is then doubled sequentially to 512 in the last block. Features extracted from the hidden layers are downsampled with global max-pooling [45] before feeding to the base classifiers.

TABLE II

FEW-SHOT CLASSIFICATION ACCURACIES (WITH STANDARD DEVIATIONS) ON *mini*-IMAGENET AND *tiered*-IMAGENET, USING A 4-LAYER CONVOLUTIONAL NETWORK AS BACKBONE. RESULTS OF THE BASELINE METHODS ARE FROM [23]. '-' INDICATES THAT THE CORRESPONDING RESULT IS NOT REPORTED IN [23].

Model		*mini*-ImageNet		*tiered*-ImageNet	
		1-shot	5-shot	1-shot	5-shot
optimization-based	Meta-LSTM [14]	43.44 ± 0.77	60.60 ± 0.71	-	-
	MAML [11]	48.70 ± 1.80	63.11 ± 0.92	51.67 ± 1.81	70.30 ± 1.75
metric-based	MatchingNet [8]	43.56 ± 0.84	55.31 ± 0.73	-	-
	ProtoNet [6]	49.42 ± 0.78	68.20 ± 0.66	53.31 ± 0.89	72.69 ± 0.74
	R2-D2 [22]	51.20 ± 0.60	$\mathbf{68.80 \pm 0.10}$	-	-
	RelationNet [7]	50.44 ± 0.82	65.32 ± 0.70	-	-
	Prototype Classifier				
	@last	53.60 ± 0.86	66.61 ± 0.73	56.78 ± 0.89	72.00 ± 0.75
	@penultimate	46.55 ± 0.84	61.42 ± 0.76	45.37 ± 0.90	60.59 ± 0.78
	@second-to-last	37.21 ± 0.86	50.02 ± 0.75	40.38 ± 0.89	53.97 ± 0.77
	SEEN (uniform)	52.32 ± 0.83	64.58 ± 0.73	53.84 ± 0.94	68.00 ± 0.75
	SEEN (weighted)	54.08 ± 0.87	66.42 ± 0.74	57.21 ± 0.93	72.21 ± 0.75
	SEEN (stacking)	$\mathbf{55.30 \pm 0.88}$	66.70 ± 0.73	$\mathbf{57.41 \pm 0.94}$	$\mathbf{72.88 \pm 0.76}$

TABLE III

FEW-SHOT CLASSIFICATION ACCURACIES (WITH STANDARD DEVIATIONS) ON *mini*-IMAGENET AND CIFAR-FS, USING A *ResNet-12* BACKBONE. RESULTS OF THE BASELINE METHODS ARE FROM [23]. '-' INDICATES THAT THE CORRESPONDING RESULT IS NOT REPORTED IN [23].

Model		*mini*-ImageNet		CIFAR-FS	
		1-shot	5-shot	1-shot	5-shot
optimization-based	SNAIL [42]	55.71 ± 0.99	68.88 ± 0.92	-	-
	AdaResNet [43]	56.88 ± 0.62	71.94 ± 0.57	-	-
metric-based	ProtoNet [6]	59.25 ± 0.64	75.60 ± 0.48	72.20 ± 0.70	83.50 ± 0.50
	TADAM [20]	58.50 ± 0.30	76.70 ± 0.30	-	-
	MetaOptNet [23]	64.09 ± 0.62	80.00 ± 0.45	72.00 ± 0.70	84.20 ± 0.50
	Prototype Classifier				
	@last	63.84 ± 0.96	74.79 ± 0.70	74.12 ± 1.01	80.86 ± 0.73
	@penultimate	65.10 ± 0.91	78.48 ± 0.63	75.61 ± 0.95	83.24 ± 0.70
	@second-to-last	60.91 ± 0.94	75.17 ± 0.68	69.91 ± 0.98	81.00 ± 0.73
	SEEN (uniform)	67.71 ± 0.91	80.05 ± 0.62	76.78 ± 0.95	84.49 ± 0.70
	SEEN (weighted)	67.73 ± 0.92	80.02 ± 0.61	76.50 ± 0.93	83.26 ± 0.67
	SEEN (stacking)	$\mathbf{67.98 \pm 0.93}$	$\mathbf{80.12 \pm 0.63}$	$\mathbf{77.11 \pm 0.97}$	$\mathbf{84.83 \pm 0.69}$

For the proposed SEEN, in both backbones, we build a prototype classifier (introduced in Section III-A) from each of the last three layers. For SEEN with stacking, the meta-learner is a 2-layer MLP with the ReLU activation. The hidden layer size is the same as the input dimensionality. After that, a softmax layer is used to guarantee the sum of combination weights equals one.

Training Strategy. During meta-training, we train the feature extractor and classifier with SGD (momentum 0.9 and weight decay of 0.0005). For all backbones and datasets, the learning rate starts at 0.1 and is reduced by 90% at epochs 10, 20, and 40. The maximum number of training epochs is 60. Following [10], [20], we set the temperature in Eqs. (1) and (4) to 10. For fair comparison, we adopt the same data augmentation procedures (random crop, color jittering and horizontal flip) as in [23], [32]. During meta-validation, we select the best feature extractor based on the 5-way 5-shot performance on the meta-validation set. For weighted averaging, we compute the weights from the average classification accuracies on the meta-validation set. For stacking, we randomly split the meta-validation set into two disjoint subsets with 60% and 40% for learning the meta-learner and early stopping respectively. We use the Adam optimizer [46] to train the meta-learner with learning rate 0.0005 and a maximum of 10 training epochs.

TABLE IV
5-WAY 5-SHOT ACCURACIES IN *mini*-IMAGENET → CUB. RESULTS OF THE BASELINES ARE FROM [31].

MAML [11]	51.34 ± 0.72
MatchingNet [8]	53.07 ± 0.74
ProtoNet [6]	62.02 ± 0.70
Cosine Classifier [3]	62.04 ± 0.76
Linear Classifier [3]	65.57 ± 0.70
Diverse-20-Full [31]	66.17 ± 0.55
Prototype Classifier	
@last	61.89 ± 0.74
@penultimate	65.58 ± 0.78
@second-to-last	64.19 ± 0.76
SEEN (uniform)	$\mathbf{67.16 \pm 0.74}$

TABLE V
ABLATION STUDY ON 5-WAY CLASSIFICATION WITH *mini*-IMAGENET.

Classifier	1-shot	5-shot
Prototype Classifier		
@last	63.84 ± 0.95	74.79 ± 0.70
@penultimate	65.10 ± 0.91	78.48 ± 0.63
@second-to-last	60.98 ± 0.94	75.17 ± 0.68
SEEN (stacking)	$\mathbf{67.98 \pm 0.93}$	$\mathbf{80.12 \pm 0.63}$
Cosine Classifier		
@last	59.21 ± 0.83	73.95 ± 0.66
@penultimate	60.74 ± 0.84	76.29 ± 0.62
@second-to-last	57.00 ± 0.86	72.56 ± 0.67
SEEN (stacking)	$\mathbf{63.12 \pm 0.86}$	$\mathbf{78.56 \pm 0.61}$

B. Few-Shot Classification

We compare the proposed SEEN with both metric-based and optimization-based few-shot methods. The metric-based methods include MatchingNet [8], ProtoNet [6], RelationNet [7], TADAM [20], R2-D2 [22], Prototype Classifier, and MetaOptNet [23]. The optimization-based methods include Meta-LSTM [14], MAML [11], SNAIL [42], and AdaResNet [43].

The experimental setup is identical to that in [23]. For baselines that do not need a meta-validation set, the meta-training and meta-validation sets are merged as a larger set for meta-training. Hence, all methods use the same amount of information.

The performance is evaluated on the meta-testing set with the 5-way k-shot ($k = 1, 5$) evaluation protocol. In each testing episode, we sample 5 classes from the meta-testing set, with $k+15$ examples from each class, where k of these are support examples for training the classifier and 15 query examples for testing. We report the average classification accuracy over 600 episodes.

Results. Table II shows results on using the shallow backbone. In the 5-way 1-shot setting, the prototype classifier built from the last layer is competitive on both datasets, while SEEN with stacking outperforms all baselines. Note that as the layer goes lower, the performance of the corresponding classifier becomes worse remarkably, suggesting that those layers contain much less discriminative features than the last layer. Hence, simply averaging the predictions from the three base classifiers is not a good strategy, and it performs even worse than the best single-layer classifier.

Table III shows results on using the deep backbone. Again, the proposed SEEN surpasses all baselines in both settings. Moreover, on *mini*-ImageNet, the deep backbone performs much better than the shallow backbone (in Table II) as expected. Among the three single-layer prototype classifiers, the best is built from the penultimate layer, which suggests that the last layer is too specific to generalize to novel classes. All the three variants of SEEN outperform the other baselines, with the one using stacking performing slightly better among the three variants.

C. Cross-Domain Few-Shot Classification

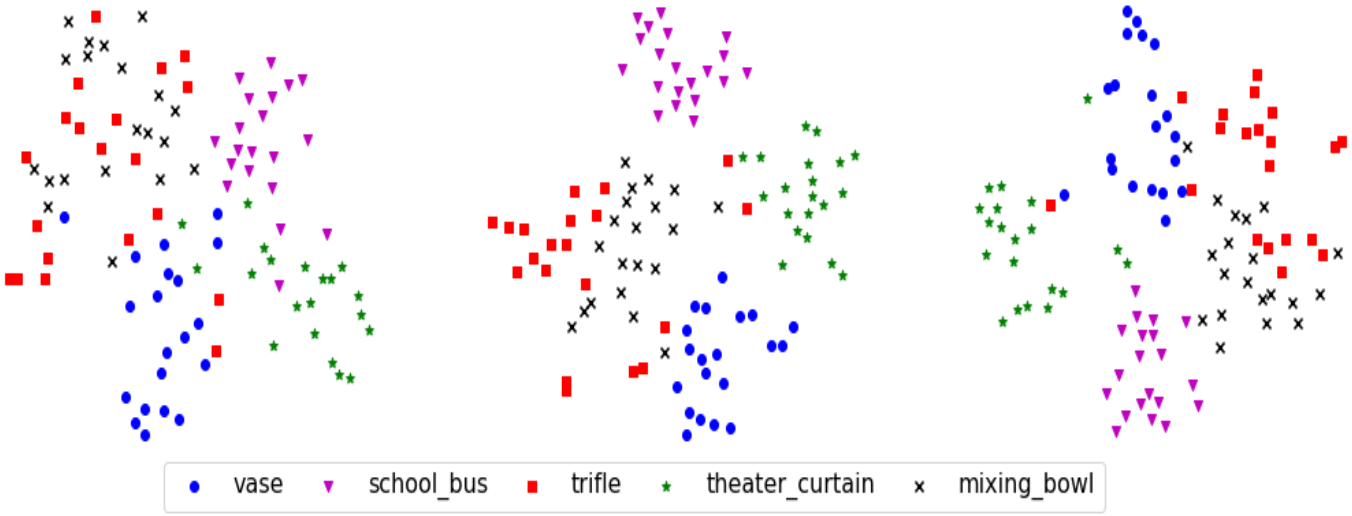
Cross-domain few-shot classification is useful in situations where we have access to a large dataset on training, but only a small number of examples from the target domain on meta-testing. In this experiment, we meta-train on the source domain *mini*-ImageNet, and meta-test on the novel target domain CUB. The setup (5-way 5-shot) is identical to that in [3], [31], and the same dataset splits and RestNet-18 [1] backbone are used. We randomly sample 1000 few-shot episodes from the meta-testing set of CUB as in [3], [31], and report the average classification accuracy.

For the proposed SEEN, we train the feature extractor using Adam with learning rate 0.001, batch size 16, and a maximum of 400 epochs. As the meta-validation set of CUB is not available, we use uniform averaging to combine predictions from the prototype classifiers built on the last three layers. We compare the proposed SEEN with the optimization-based method of MAML [11], metric-based methods (i.e., Prototype classifiers, MatchingNet [8], ProtoNet [6], cosine and linear classifier [3]), and an ensemble method (i.e., Diverse-20-Full [31]). Unlike the current state-of-the-art method Diverse-20-Full [31], which is an ensemble of 20 networks while SEEN uses only one network.

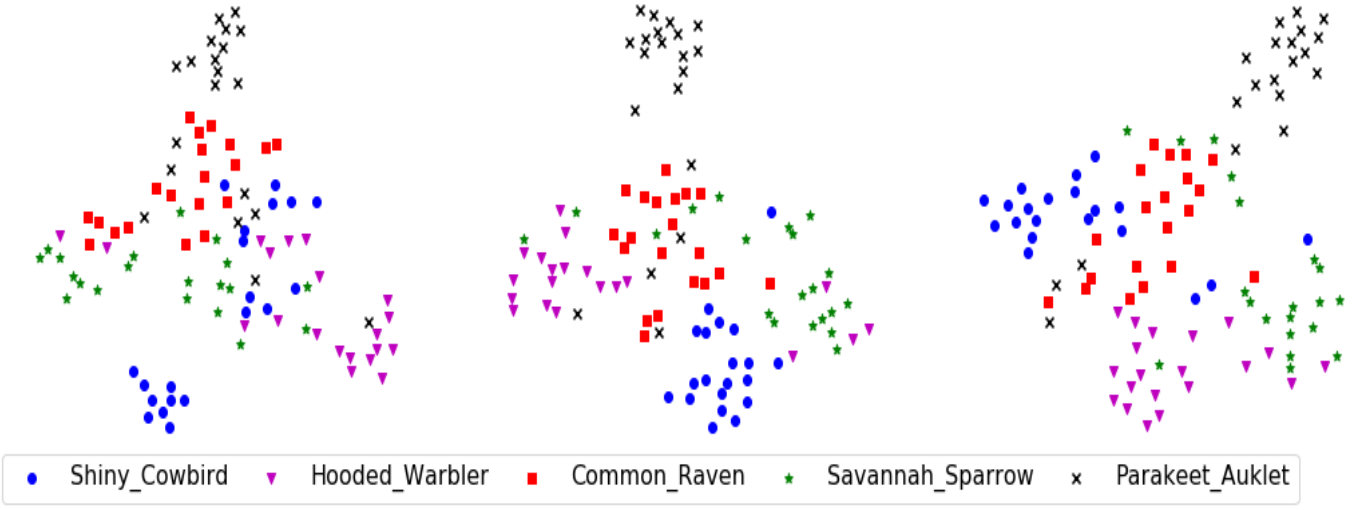
As shown in Table IV, the classifier built on the last layer performs worse than those built on the penultimate and second-to-last layers by a large margin, suggesting that the last layer is too domain-specific to generalize to the novel domain. Although none of the single-layer prototype classifiers outperforms the competitive baselines, combining them together in SEEN leads to the best performance.

D. Ablation Study

In this section, we compare the prototype classifier and cosine classifier when used with SEEN for 5-way classification on the *mini*-ImageNet dataset. The setup is the same as in Section IV-B. According to results in Table V, SEEN shows significant improvement over classifiers that use only a single layer in both 1-shot and 5-shot classification. Specifically, for the prototype classifier, combining the last three layers improves the 1-shot classification accuracy by 2.88% and 5-shot accuracy by 1.67% over the best single-layer classifier.



(a) *mini*-ImageNet → *mini*-ImageNet.



(b) *mini*-ImageNet → CUB.

Fig. 3. t-SNE visualizations of features extracted from the last three layers of the pretrained networks using *mini*-ImageNet (left: last layer; middle: penultimate layer; right: second-to-last layer). Fig. (a): 5-way 5-shot episode from the meta-testing set. Fig. (b): an episode from CUB.

For the cosine classifier, combining the last three layers improves the 1-shot classification accuracy by 2.38% and 5-shot accuracy by 2.27%. Those results show that the improvement by SEEN is agnostic to the choice of classifier. We also notice that the prototype classifier performs better than the cosine classifier, suggesting that the parameter-free classifier helps avoid overfitting in the few-shot setting.

E. Visualization

In the experiment with *mini*-ImageNet using the ResNet-12 backbone, we visualize the t-SNE [47] embeddings of the last three layers obtained from a random 5-way 5-shot episode in the meta-testing set. According to Fig. 3(a), features from the penultimate and second-to-last layers have more compact and separable structure than those from the last layer. Fig. 3(b) shows visualizations for a random 5-way 5-shot episode from the meta-testing set of CUB in the cross-domain few-shot experiment. Again, clusters in the penultimate and second-to-last layers are tighter than those from the last layer. All of those results show that the features learned in the penultimate and second-to-last layers are more discriminative and helpful for classification than those of the layer layer.

V. Conclusion

In this paper, we proposed the SEEN method for few-shot classification. We first build base classifiers on each of the last several layers and then combine them together to construct a strong model. This helps to avoid overfitting and enables the model to generalize better to novel classes or domains. Experiments on benchmark datasets demonstrate that SEEN is

an effective method to improve base classifiers. Future work may apply the proposed SEEN to other few-shot learning problems such as few-shot image segmentation.

ACKNOWLEDGMENT

This work is supported by NSFC 62076118.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[3] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2018.

[4] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[5] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–34, 2020.

[6] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4077–4087.

[7] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208.

[8] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3630–3638.

[9] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *International Conference on Learning Representations*, 2018.

[10] J. Liu, L. Song, and Y. Qin, "Prototype rectification for few-shot learning," in *European Conference on Computer Vision*, 2020.

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

[12] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," Preprint arXiv:1803.02999, 2018.

[13] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 3981–3989.

[14] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017.

[15] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few-shot learning," Preprint arXiv:1707.09835, 2017.

[16] E. Park and J. B. Oliva, "Meta-curvature," in *Advances in Neural Information Processing Systems*, 2019, pp. 3314–3324.

[17] Y. Lee and S. Choi, "Meta-learning with adaptive layerwise metric and subspace," in *International Conference on Machine Learning*, 2017.

[18] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell, "Meta-learning with warped gradient descent," in *International Conference on Learning Representations*, 2019.

[19] X. Jiang, M. Havaei, F. Varno, G. Chartrand, N. Chapados, and S. Matwin, "Learning to learn with conditional class dependencies," in *International Conference on Learning Representations*, 2018.

[20] B. Oreshkin, P. R. López, and A. Lacoste, "TADAM: Task dependent adaptive metric for improved few-shot learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 721–731.

[21] H. Li, D. Eigen, S. Dodge, M. Zeiler, and X. Wang, "Finding task-relevant features for few-shot learning by category traversal," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1–10.

[22] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *International Conference on Learning Representations*, 2018.

[23] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[24] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," Preprint arXiv:1711.04340, 2017.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[26] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *the IEEE International Conference on Computer Vision*, 2017, pp. 3018–3027.

[27] K. Hsu, S. Levine, and C. Finn, "Unsupervised learning via meta-learning," in *International Conference on Learning Representations*, 2018.

[28] S. Liu, A. Davison, and E. Johns, "Self-supervised generalisation with meta auxiliary learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 1679–1689.

[29] Y. Guo, N. C. Codella, L. Karlinsky, J. R. Smith, T. Rosing, and R. Feris, "A new benchmark for evaluation of cross-domain few-shot learning," *arXiv preprint arXiv:1912.07200*, 2019.

[30] Z.-H. Zhou, *Ensemble methods: Foundations and algorithms*. CRC press, 2012.

[31] N. Dvornik, C. Schmid, and J. Mairal, "Diversity with cooperation: Ensemble methods for few-shot classification," in *IEEE International Conference on Computer Vision*, 2019, pp. 3723–3731.

[32] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4367–4375.

[33] H. Qi, M. Brown, and D. G. Lowe, "Low-shot learning with imprinted weights," in *IEEE Conference on Computer vision and Pattern Recognition*, 2018, pp. 5822–5830.

[34] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2624–2637, 2013.

[35] 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 *International Conference on Learning Representations*, 2018.

[36] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[37] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Deep Learning Workshop*, 2015.

[38] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[39] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[40] 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 *International Conference on Learning Representations*, 2018.

[41] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[42] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.

[43] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler, "Rapid adaptation with conditionally shifted neurons," in *International Conference on Machine Learning*, 2018, pp. 3664–3673.

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, C. B. Alexander, and F.-F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[45] M. Lin, Q. Chen, and S. Yan, "Network in network," in *International Conference on Learning Representations*, 2014.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[47] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.