PMAL: Open Set Recognition via Robust Prototype Mining

Jing Lu\textsuperscript{1*}, Yunlu Xu\textsuperscript{1*}, Hao Li\textsuperscript{1}, Zhanzhan Cheng\textsuperscript{1,2†}, Yi Niu\textsuperscript{1}

\textsuperscript{1} Hikvision Research Institution, Hangzhou, China
\textsuperscript{2} Zhejiang University, Hangzhou, China
\{lujing6, xuyunlu, lihao50, chengzhanzhan, niuyi\}@hikvision.com

Abstract

Open Set Recognition (OSR) has been an emerging topic. Besides recognizing predefined classes, the system needs to reject the unknowns. Prototype learning is a potential manner to handle the problem, as its ability to improve intra-class compactness of representations is much needed in discrimination between the known and the unknowns. In this work, we propose a novel Prototype Mining And Learning (PMAL) framework. It has a prototype mining mechanism before the phase of optimizing embedding space, explicitly considering two crucial properties, namely \textit{high-quality} and \textit{diversity} of the prototype set. Concretely, a set of high-quality candidates are firstly extracted from training samples based on data uncertainty learning, avoiding the interference from unexpected noise. Considering the multifarious appearance of objects even in a single category, a diversity-based strategy for prototype set filtering is proposed. Accordingly, the embedding space can be better optimized to discriminate therein the predefined classes and between known and unknowns. Extensive experiments verify the two good characteristics (\textit{i.e.}, \textit{high-quality} and \textit{diversity}) embraced in prototype mining, and show the remarkable performance of the proposed framework compared to state-of-the-arts.

Introduction

Classic image classification problem is commonly based on the assumption of close set, \textit{i.e.}, categories appeared in testing set should all be covered by training set. However, in real-world applications, samples of unseen classes may appear in testing phase, which will inevitably be misclassified into the specific known classes. To break the limitations of close set, Open Set Recognition (OSR) (Scheirer et al. 2013) was proposed, which has two sub-goals: known class classification and unknown class detection.

Methods based on Prototype Learning (PL) obtained promising performance (Yang et al. 2018; Chen et al. 2020) recently. This group generates clearer boundaries between the known and unknowns through learning more compact intra-class feature representations using prototypes (on behalf of the discriminative features of each class). In detail, (Yang et al. 2018) learns the CNN feature extractor and prototypes jointly from the raw data and predicts the categories by finding the nearest prototypes instead of the traditional SoftMax layer. (Chen et al. 2020) advanced the framework (Yang et al. 2020) by adversely using the prototypes named reciprocal points to represent the outer embedding space of each known class, and then limiting the embedding space of unknown class. The existing methods all conduct prototype learning and embedding optimization jointly, regarding the prototypes as parameterized vectors, without direct constraint on the procedure of obtaining prototypes. Here we call them \textit{implicitly} learned prototypes, and oppositely, if imposing direct guidance on the prototypes themselves, we denote the prototypes as \textit{explicitly} learned ones. All the above-mentioned methods belong to the former category, \textit{i.e.}, the \textit{implicit} prototype-based methods. While they inevitable encounter some problems, especially in complicated situations. Two typical problems are shown in Figure 1: (1) \textbf{Undesired learned prototypes close to feature space of low-quality} samples. As in Figure 1(a), implicitly learned prototypes are mistakenly guided by the low-

\textsuperscript{1}These authors contributed equally.
\textsuperscript{†}Corresponding author.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
quality samples. As claimed in (Shi and Jain 2019), embedding of high-quality samples is discriminative while low-quality samples correspond to ambiguous features. Prototypes should represent the discriminative features of each class, so only high-quality samples are suitable. (2) Redundancy in similar prototypes and lack of diversity. Without explicit guidance, prototypes in one category show much redundant and cannot sufficiently represent the multifarious appearance. As in Figure (b), prototypes marked in green are adjacent in feature space and samples nearby show similar appearance, which implies the redundancy in learned prototypes. Besides, the airplanes in green and yellow rectangles show great distinctions, and their embeddings are located at separated positions. Obviously only using prototypes in green can not fully captures the multifarious appearance, which we require the diversity of prototype set.

Upon the above, we take high-quality prototypes and their diversity into consideration and propose to explicitly design the prototype\textsuperscript{3} mining criteria, and then conduct PL with the chosen desirable prototypes. Note that different from the existing implicit prototypes, our proposed can be regarded as explicit ones. We name the novel framework as Prototype Mining And Learning (PMAL). The framework is illustrated in Figure 2, which can be divided into two phases, the prototype mining and embedding learning orderly. (1) The prototype mining phase. High-quality candidates are first extracted from training set according to the newly proposed metric embedding topology robustness, which captures the data uncertainty contained in samples arisen from inherent low-quality factors. Then the prototype set filtering is designed to incorporate diversity for prototypes in each class. The step not only prevents the redundancy of similar prototypes, but also preserves the multifarious appearance of each category. (2) The embedding optimization phase. In this phase, given high-reliable prototypes, the embedding space is optimized via a well-designed point-to-set distance metric. The training burden is also reduced via mining prototypes in advance and feature optimization orderly, as the latter phase only work on embedding space.

Our main contributions are as follows. (1) Different from the common usage of implicitly learnable prototypes, we pay more attention on choosing prototypes with explicit criteria for OSR tasks. We point out the two important attributes of prototypes, namely the high-quality and diversity. (2) We design a OSR framework by prototype mining and learning. In the prototype mining phase, the above two key attributes are taken into consideration. In the embedding learning phase, with the chosen prototypes as fixed anchors for each class, a better embedding space is learned, without any sophisticated skills for convergence. (3) Extensive experiments on multiple OSR benchmarks show that our method is powerful to discriminate the known and unknowns, surpassing the state-of-the-art performance by a large margin, especially in complicated large-scale tasks.

Related Work
OSR is theoretically defined by Scheirer et al. (Scheirer et al. 2013), where they added an hyperplane to distinguish unknown samples from knowns in an SVM-based model. With rapid development of deep neural networks, Bendale et al. (Bendale and Boult 2016) incorporated deep neural networks into OSR by introducing the OpenMax function. Then both Ge (Ge, Demyanov, and Garnavi 2017) and Neal (Neal et al. 2018) tried to synthesize training samples of unseen classes via the popular Generative Adversarial Network. Recently, reconstruction-based (Yoshihashi et al. 2019; Ozala and Patel 2019; Sun et al. 2020) approaches are widely studied, among which Sun et al. (Sun et al. 2020) achieved promising results by learning conditional gaussian distributions for known classes then detecting unknowns. Zhang et al. (Zhang et al. 2020) added a flow density estimator on top of existing classifier to reject unseen samples. These methods all incorporate auxiliary models (e.g., auto-encoders) for OSR, thus inevitably bring extra computational cost.

Since (Yang et al. 2020; Chen et al. 2020) attempted to combine prototype learning with deep neural networks for OSR, they achieved the new state-of-the art. Prototypes refer to representative samples or latent features for each class. It is inspired by Prototype Formation theory in psychology cognition field (Rosch 1973), and is later incorporated in some deep networks, e.g., face recognition (Ma et al. 2013; Wang et al. 2016), few-shot learning (Snell, Swersky, and Zemel 2017). Yang et al. (Yang et al. 2020, 2018) introduced Convolutional Prototype Network (CPN), in which prototypes per class were jointly learned during training. Chen et al. (Chen et al. 2020) learned discriminative reciprocal points for OSR, which can be regarded as the inverse concept of prototypes. However, these methods suffer from unreliable prototypes caused by low-quality samples and lack of diversity, leading to the limited representativeness of prototypes.

Notation and Preliminaries
Notations
Let $X \rightarrow Z$ denote the mapping from input dataset $X = \{x_i\}_{i=1}^{N}$ into its embedding space $Z = \{z(x_i)\}_{i=1}^{N}$ by a trained deep classification model, where $Z \in \mathbb{R}^{N \times D}$, $N$ is the number of samples and $D$ is the embedding channel size. The feature region occupied by samples of the known class $k$ in $Z$ is referred as embedding region $Z_k$ where $k \in \{1, ..., K\}$, $K$ is the number of known classes.

![Diagram](image-url) 

Figure 2: The proposed prototype mining and learning framework. Different colors denote different classes.
Given the input $x_i \in \mathbb{R}$, the extracted feature $z(x_i)$ (simply denoted as $z_i$) is fed into the ultimate linear layer, then SoftMax operation is conducted to obtain the probability $p(z)$ of $x_i$ belonging to the $k$-th class, which is:

$$p(y_i = k|z_i) = \exp(z_i w_k + b_k) / \sum_{n=1}^{K} \exp(z_i w_n + b_n),$$

where $y_i$ is predicted class and $W = (w_1, ..., w_K) \in \mathbb{R}^{D \times K}$ and $b \in \mathbb{R}^K$ are the weight and bias term of the linear layer.

**Preliminaries of Uncertainty**

In deep uncertainty learning, uncertainties (Chang et al. 2020) can be categorised into model uncertainty and data uncertainty. Model uncertainty captures the noise of parameters in deep neural networks. What we mention in this work is data uncertainty, which captures the inherent noise in input data. It has been widely explored in deep learning to tackle various computer vision tasks, e.g., face recognition (Shi and Jain 2019), semantic segmentation (Kendall, Badrinarayanan, and Cipolla 2016) etc. Generally, inherent noise is attributed to two factors: the low quality of image and the label noise. In the scope of this work for assessing qualified samples in PL, we only regard the former. Following (Chang et al. 2020), when mapping an input sample $x_i$ into $Z$, its inherent noise, i.e., data uncertainty, contained in input will also be projected into embedding space, the embedded feature $z(x_i)$ can be formulated as:

$$z(x_i) = \phi(x_i) + n(x_i), n(x_i) \sim \mathcal{N}(0, \sigma(x_i))$$

where $\phi(x_i)$ represents the discriminative class-relevant feature of $x_i$, which can be seen as the ideal embedding for representing its identity. $\phi$ denotes the embedding model. $n(x_i)$ is drawn from a Gaussian distribution with mean of zero and $x_i$-dependent variance $\sigma(x_i)$, $\sigma(x_i)$ represents the data uncertainty (i.e., class-irrelevant noisy information caused by low quality) of $x_i$ in $Z$. The more noise contained in $x_i$, the larger uncertainty $\sigma(x_i)$ exists in embedding space. We denote $z(x_i), \phi(x_i), \sigma(x_i)$ as $z_i, \phi_i, \sigma_i$ for simplicity hereinafter.

**Prototype Mining**

Prototype mining phase has two steps orderly, the high-quality candidate selection and diversity-based filtering.

**High-Quality Candidate Selection**

Since data uncertainty captures noise in samples caused by low-quality, we exploit it for selecting high-quality samples as candidate prototypes. To model data uncertainty, a simple yet efficient algorithm is proposed, which includes the following three steps: 1) embedding space initialization, 2) data uncertainty modeling and 3) candidate selection.

**Embedding Space Initialization.** Following Monte-Carlo simulation (Gal and Ghahramani 2016), we first acquire $U$ SoftMax-based deep classifiers $\{M_u\}_{u=1}^{U}$ on the training set of known classes by repeating the training process $U$ times. Then the input data is fed into the pre-trained classifiers, obtaining $\{Z^u\}_{u=1}^{U}$. Noticing that it is sufficient to formalize different embedding space by conducting repeated training processes with random parameter initialization and data shuffling, as proved in (Lakshminarayanan, Pritzel, and Blundell 2017). Here we set $U$ to 2 for clearer illustration.

**Data Uncertainty Modeling.** Based on Sec. 3.2, the higher quality for a sample, the lower data uncertainty it has.

**Property 1.** Given a high-quality sample $x_i$, its embedding $z_i$ satisfies $z_i \approx \phi_i$.

The high-quality sample $x_i$ satisfy $\sigma_i \approx 0$, then combined with Eq. 2, we can easily obtain the above property. Suppose we select high-quality samples from training data to form the candidate prototype set $C=\{c_j^H\}_{j=1}^{H} \subset X$, where $H$ is the total candidate number. Correspondingly, their set of embedding in two different space $Z^1$ and $Z^2$ can be denoted as $\Phi^1 = \{z_i^1\}_{i=1}^{H} \approx \Phi^2 = \{z_i^2\}_{i=1}^{H} \approx \Phi^2_c = \{z_i^2_c\}_{i=1}^{H}$, where the superscript denotes the index of embedding space.

**Property 2.** Given a sample pair $(x_i, x_j), \forall i, j \in \{1, ..., H\}$, Mahalanobis distance in embedding space $Z$ can be computed by $d_{\mathcal{M}}(z_i, z_j) = \sqrt{(z_i - z_j)\Sigma^{-1}(z_i - z_j)^T}$ where $\Sigma^{-1}$ is covariance matrix. If $x_i, x_j$ are both of high quality, $d_{\mathcal{M}}(z_i, z_j)$ in different embedding space remains similar, i.e., $d_{\mathcal{M}}(z_i^1, z_j^1) \approx d_{\mathcal{M}}(z_i^2, z_j^2) \forall x_i, x_j \in C$.

**Proofs.** When only feeding the class-relevant feature $\phi_i^1$ and $\phi_i^2$ into the top linear layer of each classifier, the output probability for each category should remain consistent under the constraint of same class label $y_i$, i.e., $p(y_i = k|\phi_i^1) \approx p(y_i = k|\phi_i^2), \forall k \in \{1, ..., K\}$. Combining Eq. 1, we have the formulation:

$$\sum_{n=1}^{K} \exp(\phi_i^1 w_n^1 + b_n^1) \approx \sum_{n=1}^{K} \exp(\phi_i^2 w_n^1 + b_n^2).$$

which can be deduced to

$$\phi_i^1(\bar{w}_i^{1} - \bar{w}_i^{1}) + b_i^{1} \approx \phi_i^2(\bar{w}_i^{2} - \bar{w}_i^{2}) + b_i^{2}, \textrm{n=1, ..., K}. \ (4)$$

Averaging up all the equations for $\forall k \in \{1, ..., K\}$ leads to

$$\phi_i^1(\bar{w}_i^{1} - \bar{w}_i^{1}) + b_i^{1} \approx \phi_i^2(\bar{w}_i^{2} - \bar{w}_i^{2}) + b_i^{2}, \textrm{n=1, ..., K}. \ (5)$$

where $\bar{w} = (\sum_{k=1}^{K} w_k)/K$ and $\bar{b} = (\sum_{k=1}^{K} b_k)/K$. Taking $A = (w_1 - \bar{w}, ..., w_K - \bar{w})$ and $B = (b_1 - \bar{b}, ..., b_K - \bar{b})$, Eq. 5 can be rewritten as $\phi_i^1 A + B \approx \phi_i^2 A + B$. Given another $c_j \in C$ where $j \neq i$, the same equation $\phi_i^1 A + B \approx \phi_j^2 A + B$ can be obtained. Combining these two equations leads to

$$\sqrt{(\phi_i^1 - \phi_j^1) A^T (\phi_i^1 - \phi_j^1)^T} \approx \sqrt{(\phi_i^2 - \phi_j^2) A^T (\phi_i^2 - \phi_j^2)^T}. \ (6)$$

Here, $AA^T = (w_1 - \bar{w}, ..., w_K - \bar{w})(w_1 - \bar{w}, ..., w_K - \bar{w})^T$. As (Chang et al. 2020) pointed out, $w_n \in \{w_1, ..., w_K\}$ in $A$ can be seen as the center (or mean) of embedding space $Z_u$, i.e., $E(z_i|y_i = n) \approx w_n$. Thus $AA^T$ is a reasonable estimation for the covariance matrix $\Sigma^{-1}$ of $Z$. Consequently, Eq. 6 educes $d_{\mathcal{M}}(z_i^1, z_j^1) \approx d_{\mathcal{M}}(\phi_i^1, \phi_i^2) \approx d_{\mathcal{M}}(z_i^2, z_j^2) \approx d_{\mathcal{M}}(\phi_i^2, z_j^2)$.

**Definition 1. Embedding Topology Robustness.** Given a sample $x_i$, its relative position to other samples in embedding space $Z$ is defined by ‘embedding topology’ as:
Candidate Selection. We denote the set of all input \( x_i \) in class \( k \) as \( S_k \). To generate candidate prototype set \( C_k \) for class \( k \), we first find the sample with the highest embedding topology robustness score, i.e., \( \max \{ r(x_i) \mid x_i \in S_k \} \). Then samples with \( r(\cdot) \) value above \( \epsilon \cdot \max \{ r(x_i) \mid x_i \in S_k \} \) are elected into \( C_k \), where \( \epsilon \) is a preset threshold.

Diverse Prototype-Set Filtering

After selecting all the high-quality images into the candidate set \( C = \{C_1 \}^{K}_{k=1} \), two problems await: (1) \( C \) can be highly redundant. As in Figure 3(b), samples near \( x_1 \) share similar appearance and features. Such redundancy will bring extra computation cost in the next multi-prototype learning step. A straightforward way is to design a filtering for removing the redundant. (2) The multifarious visual appearance of object within the same class leads to distinguished feature representations. For example in Figure 3(b), \( x_1, x_2 \) and \( x_3 \) appear in different visual looking and their embedding are located at separated positions, symbolizing the diversity of embedding. Such diversity of embedding should be preserved during filtering.

Upon above, the task is turned to generate final prototype set \( P = \{P_k\}^{K}_{k=1} \) from the obtained candidate set \( C \) considering both high-quality and diversity. Specifically for each class \( k \), the method should find samples with local maximum \( r(\cdot) \) and large embedding distance to form \( P_k \), like the \( x_1, x_2 \) and \( x_3 \) in Figure 3(b).

Similar to the NP-hard coreset selection (Sener and Savarese 2018) problem, our goal is to choose \( T \) prototypes from \( C_k \) into \( P_k \) for each class \( k \). We implement it by iteratively collecting prototypes by a greedy algorithm as

\[
P_k = \bigcup_{i=1}^{T} \{ x_i \mid \min_{x_j \in C_k} d_M(z_i, z_j) > r(x_i) \}'.
\]

For initialization, we search candidates with the max \( r(\cdot) \) in \( C_k \) through \( \max \{ r(x_i) \mid x_i \in C_k \} \) to initialize \( P_k \), then append candidates satisfying Equation 8 into \( P_k \) in an iterative way. The detailed implementation is given in Algorithm 1.

Taking Figure 3(b) for example, \( x_1 \) has the max \( r(\cdot) \) thus is first elected, then \( x_3 \) and \( x_2 \) are successively added into final set, as they correspond to the largest \( d_3 \) and \( d_2 \) in \( E \). Note that Mahalanobis distance of high-quality samples remains similar in \( Z_1 \) or \( Z_2 \), thus computing \( D_k \) in either space leads to similar selected prototypes.

Embedding Optimization

Generated prototypes as anchors to represent known classes, we enlarge the distance between different embedding regions to reserve larger space for unknowns. Thus the risk of unknowns misclassified as known classes can be reduced.

Prototype-based Space Optimization

Given sample \( x_i \) belonging to known class \( m \) and \( P_k = \{p_k, t\} \), we denote the distance from \( x_i \) to prototype set \( P_k \) as \( d(z_i, z(P_k)) \), where \( z(P_k) = (z(p_{k,1}), \ldots, z(p_{k,T})) \) is the embedding of \( T \)
Algorithm 1: Filter Candidate Prototype Set with Diversity

Require: Candidate prototype set \( C = \{ C_i \}_{i=1}^K \); Class number \( K \); Prototype number per class \( T \);
Ensure: final prototype set \( P = \{ P_k \}_{k=1} \);
1: for \( k = 1 \) to \( K \) do
2: \( D_k \in \mathbb{R}^{N_k \times N_k} \) in \( Z^k \), \( N_k \) is the candidate number in \( C_k \);
3: initial a \( N_k \)-length array \( E \) with max value in \( D_k \);
4: for \( i = 1 \) to \( N_k \) do
5: for \( j \)-th candidate \( x_j \in C_k \), find its closest candidate \( x_i \) in \( C_k \), where \( r(x_j) > r(x_i) \), if exists, update \( E[i] = D_k[i,j] \);
6: end for
7: sort \( E \) in descending order, \( E_{ind} \) is the sorted index array;
8: repeat
9: add sample whose index is \( E_{ind}[0] \) in \( C_k \) into \( P_k \), then remove \( E_{ind}[0] \) from \( E_{ind} \);
10: until the number of samples in \( P_k \) exceeds \( T \)
11: end for

prototypes in \( P_k \). Then we incorporate a prototype-based constraint to optimize a better embedding space for OSR:

\[
\mathcal{L}_p = \frac{1}{N} \sum_{i=1}^{N} [d(z_i, z(P_m)) - d(z_i, z(P_u)) + \delta]_+, \tag{9}
\]

\[
\mathcal{L}_u = \arg \min_{P_u \in P \setminus P_m} (d(z_i, z(P_u))) \]

where \( P_u \) is the closest prototype set among other classes and \( \delta \) is a tunable margin. Unlike existed methods (Yang et al. 2018; Chen et al. 2020) that jointly learn sample embedding \( z_i \) and prototype representation \( z(P_k) \) in training, we update \( z(P_k) \) by directly feeding fixed prototype samples in \( P_k \) into current embedding model, thus our model can focus on learning a better sample embedding \( z_i \). Such training strategy is more advantageous since we not only avoid the unstable learning of \( z(P_k) \), but also ease training difficulty of \( z_i \). Finally, the loss in training phase is a combination:

\[
\mathcal{L} = \mathcal{L}_{cls} + \lambda_p \mathcal{L}_p, \tag{10}
\]

where \( \mathcal{L}_{cls} \) is the SoftMax loss and \( \lambda_p \) is a balancing coefficient. Besides, we design a new point-to-set distance metric with self-attention (Vaswani et al. 2017) mechanism to effectively measure the distance \( d(z_i, z(P_k)) \):

\[
d(z_i, z(P_k)) = \frac{z_i^T z^{att}(P_k)}{|z_i|^t |z^{att}(P_k)|}, \tag{11}
\]

\[
z^{att}(P_k) = \text{SoftMax}(\frac{z^T}{\sqrt{d}} z(P_k)), \quad z^{att}(P_k) = \text{SoftMax}(\frac{z^T}{\sqrt{d}} z(P_k))
\]

where \( \sqrt{d} \) is a scale factor (Vaswani et al. 2017) and \( | \cdot | \) denotes L2 norm. We use \( z_i \) to query embedding in \( z(P_k) \) to get its similarity with each prototype, and obtain weighted sum \( z_i^{att}(P_k) \). Then distance is computed by referring to similarity between \( z_i \) and all diversified prototypes, thus measure the point-to-set distance more comprehensively.

Rejecting Unknowns

Following the general routine (Yang et al. 2020), two rejection rules are adopted for detecting unknown samples: (1) Probability based Rejection (PR). We directly reject unknowns by thresholding SoftMax probability scores; (2) Distance based Rejection (DR). Unknows are rejected by thresholding the minimum point-to-set distance, i.e., \( \min_{j} \{d(z_i, z(P_k))\} \) where \( P_k \in P \), since unknown samples should have larger distance with the closest prototype set than known samples.

Experiments

Experiments on Small-Scale Benchmarks

Datasets. Following (Neal et al. 2018), we first conduct comparisons with state-of-the-arts on 6 standard datasets including (1) MNIST (LeCun et al. 1998), SVHN (Netzer et al. 2011), CIFAR10 (Krizhevsky and et al. 2009); (2) CIFAR+10, CIFAR+50: 4 non-animal classes from CIFAR10 are chosen to be known classes, then 10 and 50 animal classes are respectively sampled from CIFAR100 (Krizhevsky and et al 2009) to be unknowns; (3) TinyImageNet (TINY) (Ya and Xuan 2015); 20 classes are randomly sampled as knowns and the left 180 classes are unknowns.

Implementations. Two backbones are adopted to implement our method. The light-weighted backbone OSR (Neal et al. 2018) with parameters less than \( 1M \) is used to validate our performance when equipped on applications with limited resources. The larger-scale backbone Wide-ResNet (WRN) (Chen et al. 2020) with \( 9M \) parameters is implemented for a fair comparison with previous methods, whose parameter is still less than most existed methods (Yoshishahi et al. 2019; Oza and Patel 2019; Sun et al. 2020). We adopt Adam optimizer to train our model on each dataset for 600 epochs with batchsize 128. The learning rate starts at 0.01 and is dropped by 0.1 every 120 epochs, momentum is set to 0.9 and weight decay is 5e-4. The same optimization strategy is used for obtaining pretrained models and for embedding space optimization. For all datasets, the margin \( \delta \) is fixed to 0.5, \( \lambda_p \) is set to 1 and prototype number \( T=10 \).

Evaluation Protocols. Following (Neal et al. 2018), the evaluation includes 2 parts: (1) close set performance of known classes is reported by classification accuracy ACC on test set of knowns, and (2) unknown detection performance is evaluated by the most adopted metric AUROC (Area Under ROC Curve) (Neal et al. 2018) on the test set of both known and unknown classes. Reported results are averaged over 5 random splits. We observe two rejection rules lead to similar results, thus we simply report the results of DR.

Result Comparison. (1) Close Set Recognition: Table 1 shows we obtain the best ACC on all datasets, especially the gain reaches 2%~3% on three CIFAR datasets and TINY. We attribute it to the fact that PMAL learns more compact intra-class embedding compared to other methods (shown in Figure 6(e)~(h)), thus the classification decision boundaries among classes can be more correctly drawn. (2) Open
Table 1: Close set ACC and Open set AUROC on small datasets. ‘*’ denotes implemented results and ‘C’ is short for ‘CIFAR’.

Set Recognition: PMAL achieves the best AUROC on all benchmarks in Table 1, especially on the most complex TINY, PMAL-WRN achieves 2.2% gain compared to previous best RPL-WRN. The superiority of PMAL is more obvious when equipped on light-weight ‘OSCRI’. Compared to RPL-OSCRI, ARPL and CPN with the same backbone, PMAL-OSCRI outperforms them by a larger margin over 3.6%. Besides, the light PMAL-OSCRI only falls slightly behind PMAL-WRN, still holding top performance among all the reported results (even those with larger networks).

Experiments on Larger-Scale Benchmarks

Datasets. We further validate our method on more challenging large-scale datasets including (1) ImageNet-100, ImageNet-200 (Yang et al. 2020): the first 100 and 200 classes from ImageNet (Deng et al. 2009) are selected as known classes and the rest are treated as unknowns; (2) ImageNet-LT (Liu et al. 2019): a long-tailed dataset with 1000 known classes from ImageNet-2012 (Deng et al. 2009), and additional classes in ImageNet-2010 are as unknowns. The image number per class ranges from 5 to 1280, thus it can well simulate the problem of long-tailed data distribution.

Implementations. Similar to (Chen et al. 2020), ResNet-50 (He et al. 2016) is used as network backbone and SGD optimizer is adopted with learning rate 0.2, which drops by 0.1 every 30 epochs. Other detailed setups are the same with experiments on small-scale datasets in Section 4.

Result Comparison. The same metrics (ACC and AUROC) are used for evaluation. See Table 2, our method improves close set ACC by 3.2% and 2% on ImageNet-LT and ImageNet-200. Moreover, open set AUROC is significantly enhanced by 16.5%, 12.6% and 13.7% compared to state-of-the-art. Such performance gains are much more evident than the improvements on small-scale datasets, reflecting that our method possesses larger advantage in more challenging large-scale tasks. It can be observed that the parameter number (apart from the adopted same backbone) of previous methods increases along with known class number, which exceeds a non-negligible cost 2M on ImageNet-LT. The increased prototype parameters may aggravate the difficulty of model training, thus deteriorate their performance. Instead, since PMAL brings no parameters for prototypes, its performance is invariant to the number of known classes, which explains the promotion on more complicated datasets.

Besides, the huge promotion on long-tailed dataset shows PMAL can better handle the problem than the existing. Previous ones tend to learn unreliable prototypes for those few-sample classes, due to unbalanced training of various classes. While PMAL directly mines prototypes from training data, thus produces stable prototypes with fewer samples.

Detailed Analysis

Ablation of Each Component in PMAL. As shown in Table 3, we perform each experiment of the proposed components for ablation study, including the components regarding the two properties, i.e., high-quality and diversity (see Sec. 4), and the embedding learning procedure using the point-to-set distance metric (see Sec. 5.1).

- In prototype mining (PM) phase, both high-quality and diversity matters in prototypes, where high-quality is the more crucial factor validated by our experiments. Jointly combining the two further boosts the performance.
- In embedding optimization (EO) phase, no ‘OC’ denotes the commonly adopted way: computing the distance
Table 2: Comparisons on 3 large-scale datasets. We denote ‘ImageNet’ as ‘IN’ for simplicity.

<table>
<thead>
<tr>
<th>Components</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-Quality</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Diversity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>EO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point-to-Set</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AUROC</td>
<td>80.3</td>
<td>78.1</td>
<td>81.6</td>
<td>80.2</td>
<td>81.9</td>
<td>83.1</td>
</tr>
</tbody>
</table>

Table 3: Ablations of each module on TinyImageNet.

Table 4: Comparisons with other methods on the quality and diversity property.

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)Randomization</td>
<td>81.5</td>
<td>79.1</td>
</tr>
<tr>
<td>(b)Clustering</td>
<td>81.8</td>
<td>79.6</td>
</tr>
<tr>
<td>Ours</td>
<td>84.7</td>
<td>83.1</td>
</tr>
</tbody>
</table>

Table 5: AUROC under different hyper-parameters, including $T, \varepsilon, \alpha, \beta, \delta$.

Effect of Diversity-based Filtering. (1) Number of prototypes: We vary prototype number $T$ to quantify the effect of diversity on TinyImageNet. As in Table 5, diverse prototypes lead to much better performance than a single prototype, we obtain the best result when $T=10$. When $T$ increases over 10, performance gradually stabilized. To interpret the advantage of diversity, we visualize embedding space when $T=1$ or $T=10$ in Figure 5(a). Evidently, the chosen 10 prototypes appear in diverse visual looking and their embeddings are located at separate positions. Moreover, using 10 prototypes learns more compact intra-class embedding regions, where unknown sample points are farther from the embedding region of known classes. The reason can be illustrated by Figure 5(b): when $T=1$, sample $s_B$ from class
PMAL is capable of mining trustworthy prototypes and optimizing embeddings in more challenging tasks.

Hyper-parameters. (1) Threshold $\epsilon$. $\epsilon$ controls the balance of prototype quality and diversity. A larger $\epsilon$ implies more strict condition to ensure quality, but less samples will be elected as candidates, thus diversity among candidates is reduced. A smaller $\epsilon$ is on the contrary. As in Table 5, setting $\epsilon$ too small or large both results in AUROC declined. We find $\epsilon$ in the range [0.6, 0.8] leads to stable performance. (2) Initial model number $U$. Optionally we can adopt more than 2 models to compute $r(\cdot)$, that is, we first compute $r(\cdot)$ with Equation 7 between two arbitrary models then average the results. AUROC remains similar as we increase $U$, shown in Table 5, which implies 2 models are already sufficient to extract effective prototypes. Note that we adopt 2 models for prototype mining, but only one model is employed for prototype learning, thus no extra model parameters are added during inference. (3) Margin $\delta$ and $\lambda_p$. $\delta$ decides the separability of different embedding regions. When $\delta$ is too small, different regions can not be well separated. But if $\delta$ becomes too large, $L_p$ will grow to a large value overwhelming $L_{cls}$, causing $L_{cls}$ hard to converge. See Table 5, a value between 0.3 and 0.5 for $\delta$ can produce better AUROC results. For loss weight $\lambda_p$, when we vary it among 0.5, 0.8 and 1, the resulted AUROC are 82.6, 82.8 and 82.9, which implies PMAL is not very sensitive to $\lambda_p$. For all the validations in various tasks, we adopt the universal hyper-parameter setting: $T=10$, $\epsilon=0.7$, $U=2$, $\delta=0.5$ and $\lambda_p=1$.

Conclusion
This paper proposes a novel prototype mining and learning algorithm. It directly discovers high-quality and diversified prototype sets from training samples. Then with generated prototypes, the OSR model can focus on optimizing a better embedding space in which known and unknown classes are separated. Extensive experiments on various benchmarks show that our method outperforms the state-of-the-art approaches. In future work, we will explore our prototype mining mechanism in broader tasks other than OSR.
References