Multi-stage Deep Classifier Cascades for Open World Recognition

Xiaojie Guo
George Mason University
Fairfax, VA
xguo7@gmu.edu

Amir Alipour-Fanid
George Mason University
Fairfax, VA
aalipour@gmu.edu

Lingfei Wu
IBM Research AI
New York
wuli@us.ibm.com

Hemant Purohit
George Mason University
Fairfax, VA
hpurohit@gmu.edu

Xiang Chen
George Mason University
Fairfax, VA
xchen26@gmu.edu

Kai Zeng
George Mason University
Fairfax, VA
kzeng2@gmu.edu

Liang Zhao
George Mason University
Fairfax, VA
lzhao9@gmu.edu

ABSTRACT

At present, object recognition studies are mostly conducted in a closed lab setting with classes in test phase typically in training phase. However, real-world problem are far more challenging because: i) new classes unseen in the training phase can appear when predicting; ii) discriminative features need to evolve when new classes emerge in real time; and iii) instances in new classes may not follow the "independent and identically distributed" (iid) assumption. Most existing work only aims to detect the unknown classes and is incapable of continuing to learn newer classes. Although a few methods consider both detecting and including new classes, all are based on the predefined handcrafted features that cannot evolve and are out-of-date for characterizing emerging classes. Thus, to address the above challenges, we propose a novel generic end-to-end framework consisting of a dynamic cascade of classifiers that incrementally learn their dynamic and inherent features. The proposed method injects dynamic elements into the system by detecting instances from unknown classes, while at the same time incrementally updating the model to include the new classes. The resulting cascade tree grows by adding a new leaf node classifier once a new class is detected, and the discriminative features are updated via an end-to-end learning strategy. Experiments on two real-world datasets demonstrate that our proposed method outperforms existing state-of-the-art methods.

KEYWORDS

Open-world recognition; deep neural networks

ACM Reference Format:
Xiaojie Guo, Amir Alipour-Fanid, Lingfei Wu, Hemant Purohit, Xiang Chen, Kai Zeng, and Liang Zhao. 2019. Multi-stage Deep Classifier Cascades for Open World Recognition. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357981

1 INTRODUCTION

In recent decades, the development of machine learning and pattern recognition techniques has enabled many different kinds of objects to be learned, detected and analyzed for practical applications [13, 16, 25, 35, 43, 44]. However, although most recognition systems are designed for a static closed condition, assuming that all the classes are known in the training phrase [3, 37], the real world is an **open set environment**, where only a small part of the entire set of objective classes is known. This requires the recognizer to be able to reject unknown/unseen classes that appear during the inference stage, which has been addressed by several researchers [3, 24]. A real-world application should not only be able to infer but also to classify these newly emerging classes. For instance, in a device recognition problem using device signal fingerprinting in the area of cyber-security, the abundant device types make it impossible to include all the possible data types in the training phase. The example in Fig. 1 demonstrates what happens when there may be four known types of devices in the training phase and three new types that are not seen in the training phase but are present in the prediction phase. Here, the recognizer must not only be able to distinguish the four known devices and be aware of the existence of new types, but must also learn to classify these unknown devices during test phase. To address the above task, the **Open World Recognition** (OWR) (which is also named as Open World Learning) problem has been formalized. This is composed of three tasks: the classification of known classes, the detection of unknown classes, and updating the model to learn new classes in an environment where new classes come continuously [2].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM '19, November 3–7, 2019, Beijing, China
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357981

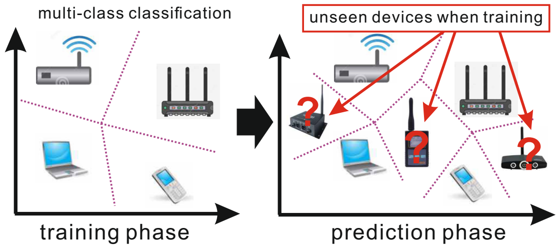


Figure 1: In recognizing the device type through the emitted signal fingerprinting, going beyond conventional multi-class recognition on four known devices, three unseen/new devices must be continuously identified and distinguished in the prediction phase.

As a new domain, OWR is starting to attract increasing attention, with the limited number of existing works being divided into two categories. The first category assumes that a batch of instances from new classes emerge at about the same time. For instance, the time series being divided into several instances can appear together. Two metric-based learning models have been proposed to detect unknown classes and include new classes using the Nearest Class Mean [2, 39]. Another category assumes that instances of new classes come continuously, and that the misclassified instances may then have a critical impact on the subsequent learning. Thus, an incremental random tree [23] with a buffer to store the emerging instances from new classes has been proposed; an alternative approach that has been suggested is based on a heuristic recognition model with side-information for the emerging classes [19]. Both these categories require side-information about which instances are from the same classes (e.g., for Twitter topic recognition, tweets with the same hashtags are from the same topics). However, all the methods proposed so far for open-world recognition are highly dependent on pre-defined handcrafted features and is consequently difficult to learn unique and inherent features in real-time as new classes arise. Furthermore, handcrafted features do not necessarily include discriminative and critical features. For example, the device type recognition problem based on signal fingerprinting in Fig. 1 is very difficult since the contents carried in the signals are varied and mixed with ambient noise. This makes it hard to discover any distinguishable features buried in the raw signals purely based on human domain knowledge.

Currently, the end-to-end OWR problem, where the features of new classes need to be automatically extracted and included, cannot be handled by the existing techniques due to the following challenges: 1) **It is difficult to update features for an end-to-end model whose architecture is predefined**. The involvement of new classes incurs the need to learn more and new features. However, the architectures proposed in existing studies are pre-defined and cannot be updated to include new classes. Retraining a model with an expanded architecture to update its features is time-consuming and may result in catastrophic forgetting problems [12], where the already learned features are replaced by the newly aquired ones. 2) **Features learned based on existing classes are not sufficient to detect future unknown classes**. Newly updated features should also be able to detect future unknown classes whose instances are not available when updating the current model. Even though we assume that the features can be updated with the new classes, they may still be out-of-date when it comes to distinguishing future potential new classes. 3) **It is difficult to utilize the relationships among instances from the same class**. Since instances in the same classes may not be independent and identically distributed, enforcing the feature similarities is critical but challenging for end-to-end supervised learning, which is commonly conducted without considering the relationships among instances.

To address the above challenges and explore the open world recognition problem based on end-to-end learning, we propose a novel Multi-stage Deep Classifier Cascades (MDCC) architecture that leverages a generic cascade architecture to incrementally learn the new class in an end-to-end fashion without retraining the whole model. A one-class classifier that is trained based on a dynamic reference set is proposed to include new classes as well as detect potential future unknown classes to address the second of the above challenges. A self-describing regularization is also proposed to utilize the relationships among instances to address the third of the above challenge. The proposed MDCC has the following distinct features.

- **The development of a new end-to-end framework for open world recognition**. To the best of our knowledge, the proposed MDCC is the first generic framework to address the open world problem based on automatic feature extraction utilizing a novel classifier cascade for identifying multiple classes.
- **The proposal of a one-class classifier based on a dynamic reference set**. One-class classifiers are proposed and leveraged as leaf nodes that not only automatically learn new features but also detect any potential future unknown classes. Using the proposed dynamic reference set to train the one-class classifier ensures the scalability of the computation and the descriptiveness of the extracted feature.
- **The proposal of a self-describing regularization that utilizes the relations among the instances**. In order to enhance the learning of the features of the new classes in the leaf nodes and narrow the feature spaces of the new classes, a new regularization is proposed by minimizing the inconsistency of features among the related instances.
- **Extensive experiments to validate the effectiveness of the proposed model**. Extensive experiments on two real-world datasets demonstrates that MDCC is capable of incrementally detecting and learning emerging new classes, significantly outperforming comparison methods.

2 RELATED WORK

Open set learning: Open set recognition was first introduced in [31], which considers the problem of detecting unseen classes that are never seen in the training phase [20, 41]. Many open-set recognition methods based on SVM [4, 22] and NCM [3] have since been proposed, but all built on shallow models for classification. Scheirer et al. [31] formulated the problem of open set recognition for static one-vs-all learning scenario by balancing open space risk while minimizing empirical error,going on to extend the work to multi-class settings by introducing a compact abating probability model [32]. For the scalability problem, Fragoso et al. [11] proposed the use of a scalable Weibull based calibration for hypothesis generation to model matching scores, but did not address its use for the general recognition problem. Bendale et al. [3] proposed a novel detection method dealing with deep model architecture by introducing an openmax layer, while Perera et al. [27] proposed a one class classification based on the DCNN which can be used as a novel detector and outlier detector for a single known class. However, none have not addressed the problem of how to incrementally update their model after a new class has been recognized.

Incremental learning: Incremental learning refers to a continuous learning process with new data that has been labeled [17, 29, 36, 42]. Many incremental methods are based on SVM [5, 17] and random tree methods [29]. Cauwenberghs et al. [5] proposed an incremental binary SVM by means of saving and updating KKT conditions, while Yeh et al. [40] extended this approach to include object recognition and demonstrated multi-class incremental learning. However, incremental SVMs suffer from multiple drawbacks. The update process is extremely expensive (quadratic in the number of training examples learned) and depends heavily on the number of support vectors [15], so Rebuffi et al. [29] proposed a memory-controlled training strategy that learns the coming classes as well as maintains the training load. Goodfellow et al. [12] addressed the knowledge transfer concept in neural network incremental learning. Inspired by this, an error-driven based convolution neural network was introduced by Xiao et al. [38], whose model is extended like a tree structure to avoid the catastrophic forgetting problem. Rusu et al. [30] proposed a progressive network that retains a pool of pretrained models in training and then learns lateral connections from these to extract features for new tasks. However, all these multi-class incremental learning methods and incremental classifiers are incremental only in terms of additional training samples, not additional training categories. Thus the representation drift of incremental learning is not a problem considered in our problem, where each class has a true and static description over time [2, 7].

Scalable Learning: different from incremental learning problem, other researchers have proposed tree based classification methods to address the scalability of object categories in large scale visual recognition challenges [8, 10, 18, 21]. Recent advances in the deep learning domain[14, 34] of scalable learning have resulted in state of the art performances, which are extremely useful when the goal is to maximize classification/recognition performance. These systems assume a priori availability of comprehensive training data containing both images and categories. However, adapting such methods to a dynamic learning scenario becomes extremely challenging. Adding object categories requires retraining the entire system, which could be unfeasible for many applications. As a result, these methods are scalable but not incremental.

Open world recognition: Open world recognition considers both detection and learning to distinguish the new classes. Bendale et al. proposed a NCM learning algorithm that relies on the estimation of a determined threshold in conjunction with the threshold counts on some known new classes [2]. For a more practical situation, Rosa et al. proposed an online-learning approach that involves the NBC classifier instead of NCM [7], while Mu et al [23] proposed an online learning for streaming data where new classes come continuously. It is worth noting that Bayesian non-parametric models [1, 9] are not related to our problem. Though they were originally proposed to identify mixed components or clusters in the test data that may cover unseen classes, their clusters are not themselves classes and multiple clusters must be mapped to one class manually.

Table 1: Important notations and descriptions

Notations	Descriptions
X_n	Representation of the nth coming instance
Y_n	True Label of the nth coming instance
Y_n'	Predicted label of the nth sample of coming instances
C_t	Class of the tth leaf model
B	Size of Collection set $\mathcal{B}$
H	Length of the extracted feature vector of each instance
F	Size of reference set
$\mathcal{K}_t$	Number of known classes at Stage t
$\mathcal{B}$	Collection set of newly detected instances
$\mathcal{F}$	Reference set with fixed size
θ	Threshold for selecting rejection line for leaf node

3 PROBLEM FORMULATION

This paper focuses on the new problem of end-to-end open world recognition (OWR), where instances with various classes arrive continuously and must be recognized. Once a new class is detected, the multi-class recognition model should incrementally learn to accept/classify instances with this class. The following argument provides the notations and the mathematical problem formulation.

As new classes appear continuously and must be detected and incorporated accordingly, we define an OWR process consisting of several stages, with a new class detected and learned at each stage. At a Stage t, let the classes that are included in the recognizer as known classes be labeled by the positive integers $\mathcal{K}_t = \{1, 2, ..., C_t\} \subseteq \mathbb{N}^+$. Instances from either known or unknown classes are represented as (X_n, Y_n), where X_n is the nth instance of all the incoming instances and could be represented as either a matrix (e.g., an image) or a vector (e.g., a time series of objects). $Y_n \in \mathcal{K}_t$ if X_n is from known classes, and $Y_n = 0$ if X_n is from unknown classes. In this scenario, some instances come with an identity function I which provides side-information indicating whether two instances are from the same class or not, i.e., $I(X_n, X_m) = 0$ *or* 1, where 1 indicates that instance X_n and X_m belong to the same class and 0 that they do not. The open world recognition problem contains three tasks: classifying the known classes, detecting newly emerging classes and learning the new classes. Therefore, the problem is defined as learning a dynamic recognition function $M_t : X_n \rightarrow Y_n \subseteq \mathbb{N}, \ t = 1, \cdots$ which contains unique feature patterns that are used to recognize known classes and detect unknown classes. Furthermore, the recognition function M_t should be scalable to recognize all the classes that have ever been seen so that at each stage it is possible to scalably update the recognition mapping from Stage t to Stage $t + 1$ as: $M_t \rightarrow M_{t+1}$ using the dynamic features to include the new class.

However, to handle all the solutions to the end-to-end OWR problem, several challenges must be considered: 1) It is difficult when performing an end-to-end feature extraction in a fixed architecture to both include the features for a newly detected class and maintain the previously learned features for distinguishing known classes, especially considering the memory and time consumption. 2) It is difficult to ensure the updated features do indeed distinguish the newly added class from the potential future unknown classes. 3) It is difficult to utilize the relationships among the instances indicated by I to assist the feature extraction and maintain the consistency of feature patterns for instances with the same class.

4 THE PROPOSED MDCC

In this section, we propose the Multi-stage Deep Classifier Cascades (MDCC) to address the above challenges in the open world recognition scenario. First, an introduction of the overall architecture and the incremental process of the cascades are given. Then, a more detailed consideration of the proposed deep one-class classifier and a self-describing regularization mechanism are presented.

4.1 Overall Architecture

Multi-stage cascade architecture. To incrementally detect and learn new classes for the open world problem, we propose a novel Deep Convolution Neural Network (DCNN) based cascade architecture which consists of a root and several leaf classifiers trained in an end-to-end fashion. It can either reject the instance as a new class or classify the accepted instance as a member of the known classes. The DCNN is chosen as the base model due to its excellent performance and general ability to deal with input instances in any form. The merits of the overall recognition model are as follows: 1) It can contain sufficient and unique features for distinguishing among all the known classes at any stage. 2) It can increment the leaf nodes to both recognize a newly added class as well as detect future unknown classes. 3) It can learn and include new features efficiently without disturbing the existing features.

Fig. 2 shows the overall architecture of the proposed MDCC. At Stage 0, the function M_0 is realized by a root node f_0 to classify the existing known classes and detect unknown classes, as shown in the orange rectangle in Fig. 2. At Stage t, once an unknown class is detected by the function $M_t = [f_0, ..., f_t]$ and sufficient instances for this new class are collected in a buffer, a one class classifier f_{t+1} is trained to learn the feature information of the new class and it is added as a leaf node of the cascade. The recognition function is then updated as $M_{t+1} = [f_0, ..., f_{t+1}]$. The loss minimized at Stage t is defined as follows:

$$\mathcal{L}(M_t) = \begin{cases} \mathcal{L}_{root}(f_0) & t = 0 \\ \mathcal{L}_{leaf}(f_t) & t > 0, \end{cases} \tag{1}$$

where $\mathcal{L}_{root}(f_0)$ is the cross entropy loss for training the root classifier, and $\mathcal{L}_{leaf}(f_t)$ is the overall loss function of the leaf classifier f_t, which is described in more detail in Section.4.2. Optimization methods (e.g., Stochastic gradient descent and the Adam algorithm) based on Back-propagation technique can be utilized to optimize each node model.

To recognize an instance X_n at Stage t via the function $M_t = [f_0, ..., f_t]$, X_n is input into each of the node models in turn until it is accepted by a node. The input of each leaf node is the instance rejected by the previous node. Specifically, at root node f_0, if X_n is accepted as known, it will be classified by f_0; Otherwise, it enters the next node. At a leaf node $f_i (i < t)$, if X_n is accepted, it is predicted as Class C_i; Otherwise, it enters the next node. If X_n is finally rejected by the last node f_t, then it is identified as being of unknown class at stage t and is stored in a buffer. All the instances detected as new classes will be assigned into the various buffers based on their identity function. Thus, each buffer $\mathcal{B}$ contains the instances of the same class with which to train a leaf node. The detailed process for the open world recognition of instance X at Stage t are presented in Algorithm. 1.

Algorithm 1 Single instance recognition process for open world recognition at Stage t.

```
Input: The incoming instance $X_n$
Requires: Current cascade model $M_t = [f_0, ..., f_t]$.
Output: The predicted label $Y'_n$ of $X_n$.
Let $i = 0$.
while $i <= t$ do
    $Y'_n = f_i(X_n)$ .
    if $Y'_n \in \mathcal{K}_t$ then
        Return $Y'_n$.
    else
        i=i+1
        Continue.
    end if
end while
if i>t then
    Return $Y'_n = 0$
    Train leaf node $f_{t+1}$
    Update $M_t$ to $M_{t+1} = [f_0, ..., f_{t+1}]$
end if
```

Root node at initial stage. To extract features and classify the multiple known classes at the initial stage, a DCNN is utilized for the multi-class classification as the root node. This DCNN consists of several convolution layers, average pooling layers, and fully connected layers, as well as the SoftMax layer used to recognize the known classes. In addition, to detect any new classes, the openmax layer proposed by [3] is utilized in parallel with the softmax layer, as shown in Fig.2 (a). This openmax layer helps to decide whether to accept the instance as a known class or not and, once accepted, the instance is then classified by the softmax layer. Specifically, the softmax layer outputs the probability of the instance belonging to each known class $P(Y'_n = k|X_n, k \in \mathcal{K}_0)$. While the openmax layer outputs the probability of X_n being the unknown class as $P(Y'_n = 0|X_n)$.

The output of the openmax layer is computed based on the distances between the output of the activation layer (the last layer before the SoftMax layer) and $\mu_k (k \in \mathcal{K}_0)$ for each class k through a Weibull model. First, μ_k is computed during the training phrase as the mean of the activation outputs of all the training instances belonging to class k. Then a Weibull model obtained through libMR [6] is used to model the distribution of μ_k and the largest distance between μ_k and all the correctly classified samples. Based on the Weibull model generalized, we can compute the Weibull CDF probability on the distances between $P(Y'_n = 0|X_n)$ and μ_k. Since the probability is expected to be meaningful only for a few top ranks,

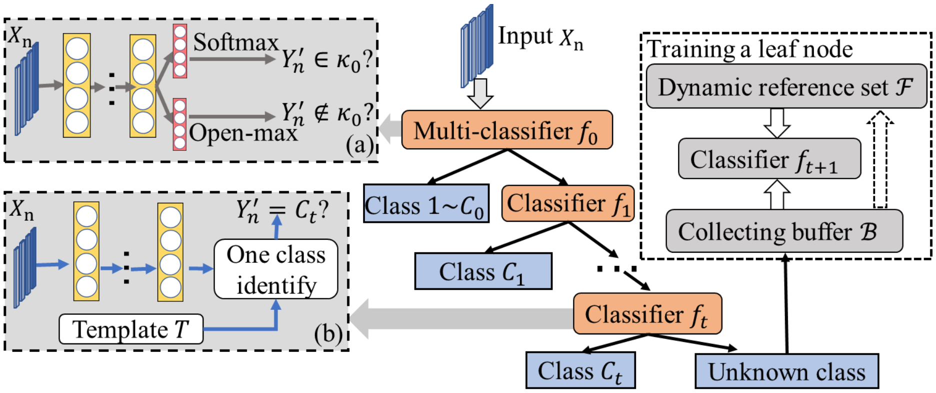


Figure 2: Architecture of Multi-stage Deep Classifier Cascades and model details of (a) root node and (b) leaf nodes

we compute the weights for the top ranking classes and use these to scale the Weibull CDF probability. The $P(Y'_n = 0|X_n)$ will be revised based on the changed scores. For the new class, a pseudo-activation layer is computed to keep the total $P(Y'_n = 0|X_n)$ value constant. The details of the Weibull model and openmax computation can be found in [3, 33].

Leaf nodes with dynamic reference set. We propose to train a deep one-class classifier that both recognizes the newly detected class and detects the future new classes. As each new class is detected, it is necessary to update the current recognizer to include this new class without retraining the model. The one-class classifiers only need to learn the features of the new class and can reject all the instances not belonging to this class. To learn the features that not only describe the new class but also distinguish it from future unknown classes, each one-class classifier incorporates an end-to-end feature extraction part and a distance-based classifier. This one-class classifier is trained based on a proposed dynamic reference set in order to both maintain the performance and minimize memory consumption.

Self-describing regularization. In the one-class classifier training, to further minimize the feature space required to largely distinguish the detected new class from others and fully utilize any side-information on the instances' relationships, we propose a self-describing regularization by enforcing the feature consistency of collected instances in the buffer. The instances are collected based on the side-information they contain regarding the identity function indicating the relationships among the instances of unknown classes.

4.2 Deep one-class classifiers as leaf nodes

To learn the features required to describe the newly detected class, we propose the one-class classifiers as leaf nodes trained in an end-to-end fashion, as shown in Fig. 2 (b). The leaf nodes in the MDCC are incrementally added at each stage to learn and include the new class' information, simultaneously detecting any future unknown classes simultaneously. The one-class classifier consists of two parts: a feature extractor and a distance-based classifier. For convenience, here the newly detected class is named the *target class* for training a classifier.

End-to-end Feature extraction. The feature extraction part of a leaf node f_t is based on a DCNN model g_t at Stage t, which is trained by improving the outer-describing property of the feature space. The outer-describing property refers to the ability of features to fully describe various classes. Improving the model's outer-describing ability can minimize the feature space needed to describe the target class and, thus, improve the distinguishability of the target class from other unknown classes. Specifically, a reference set $\mathcal{F} = \{(X_n^{(\mathcal{F})}, Y_n^{(\mathcal{F})})\}_{n=1}^F$ consisting of instances from various existing known classes is utilized to enlarge the outer-describing ability for describing the newly detected class at each stage. F is the number of instances in $\mathcal{F}$. The reference set first collects instances of each known class at Stage 0 and then continues to included instances of more classes at each stage. The outer-describing loss for training g_t can thus be minimized as follows:

$$\mathcal{L}_{outer} = \mathbb{E}_{X_n^{(\mathcal{F})}, Y_n^{(\mathcal{F})}}[-\sum_{n=1}^{F} log(g_t(X_n^{(\mathcal{F})}))]. \quad (2)$$

During the testing phase, the features for one instance can be extracted from the layer before the softmax.

Error-corrective distance based classifier. An error-corrective distance based classifier is proposed for the extracted features to evaluate whether an instance is from the target class or not. Since the features extracted by g_t only have the ability to describe various classes, its softmax layer cannot distinguish the new class from any future unknown classes. To deal with this, each instance is evaluated based on the distance $D(h_t(X_n), v_t)$ between its features $h_t(X_n)$ and the mean feature vector v_t of the training instances in the buffer $\mathcal{B} = \{(X_n^{(\mathcal{B})}, Y_n^{(\mathcal{B})})\}_{n=1}^B$, where B refers to the number of instances. $h_t(X_n)$ is the feature vector outputted from the feature extraction layers (before the softmax layer) in g_t. v_t is computed as $\sum_{n=1}^B D(h_t(X_n^{(\mathcal{B})}), v_t)/B$. The distance function D can be any distance measurement, for example the cosine distance. Then the maximum distance $d_t = max\{D(h_t(X_n^{(\mathcal{B})}), v_t)\}_{n=1}^B$ based on the training instances in buffer $\mathcal{B}$ is used as the rejection line. In addition, to minimize the influences of errors generated from the last leaf node's recognition, we allow θ percent of instances to be outlier instances and filter out the largest θ percent of distances, selecting the rejection line as $d_t^{(\theta)}$. Thus, the predicted label for an object X_n in the leaf node f_t is computed as follows:

$$f_t(X_n) = \begin{cases} unknown & \text{if} \quad D(h_t(X_n, v_t) >= d_t^{(\theta)} \\ C_t & \text{if} \quad D(h_t(X_n, v_t) < d_t^{(\theta)}. \end{cases}$$

Dynamic Reference Sets. As new classes arrive continuously, to maintain the diversity of the outer-describing property and limit memory consumption, a dynamic reference set with a fixed size rather than one that is incrementally enlarged is proposed. On one hand, the more classes that are contained in the reference set for training, the better outer-describing ability the feature space will have, so after a new leaf node is built, the instances from the newly added class need to be included in the dynamic reference set. However, on the other hand, to avoid a rapid escalation in the computation and memory costs due to the increasing size of the reference set, we must restrict the total number of instances in the set by reducing the number of instances of each class. The number of instances for each class in the reference set is computed as: F/C_t, where F is the fixed size of the reference set. The reduced instances for each class are randomly chosen.

4.3 Self-describing regularization

Feature space regularization is proposed due to two considerations: 1) to improve the ability of the extracted features to describe the target class, which is referred to as its self-describing property; and 2) to remedy the deteriorating performance of end-to-end feature extraction due to the decresing number of of instances of each class in the dynamic reference set. As instances of new classes are mixed, instances from the same new class must be minimized and instances from different classes are maximized. Thus, for any two instances X_n and X_m detected as unknown classes, the distance between their features should be minimized as: $min\ I(X_n, X_m)||h_t(X_n) - h_t(X_m)||_2$, where I is the pre-knowledge indication of the relationships between two instances. Since several instances with the same class are collected in $\mathcal{B}$ with the property of $I(X_n^{(\mathcal{B})}, X_m^{(\mathcal{B})}) = 1$, the self-describing regularization for training a one-classifier is minimized, which is expressed as follows:

$$\mathcal{R}(X_n) = \frac{1}{BH}\sum_{n=1}^{B}(h_t(X_n^{(\mathcal{B})}) - \omega_t)^T(h_t(X_n^{(\mathcal{B})}) - \omega_t),$$

where H is the length of the extracted feature and ω_t is the mean feature vector of all the instances' features in $\mathcal{B}$. Then the overall loss function of a leaf node f_t is computed as:

$$\mathcal{L}_{leaf}(f_t) = \mathbb{E}_{X_n^{(\mathcal{F})}, Y_n^{(\mathcal{F})}}[-\sum_{n=1}^{F} log(h_t(X_n^{(\mathcal{F})}))]$$
$$+\beta\frac{1}{BH}\sum_{n=1}^{B}(h_t(X_n^{(\mathcal{B})}) - \omega_t)^T(h_t(X_n^{(\mathcal{B})}) - \omega_t),$$

where β controls the trade-off between the $\mathcal{L}_{outer}$ and $\mathcal{R}(X_n)$. For optimization, the gradients minimizing the two components of the loss function are computed separately.

It is worth noting that the side-information requirement is not a limitation of the proposed MDCC. First, the existing works typically ignore this assumption and implicitly assume the new classes are unmixed or the samples arrives in a batch. Instead, our method utilizes this prior knowledge by proposing a self-describing regularization and yields better experimental results, as shown in the next section. Second, from another point of view, the proposed MDCC could be categorized as multi-instance learning based on knowing the relationships among the samples. Thus, MDCC is a general method and can include the case of single-instance learning without the need for an indicating function.

5 EXPERIMENT

In this section, the effectiveness of the proposed MDCC is evaluated for two real-world datasets and the results obtained compared with those of three existing state-of-the-art method in the domain of open-world recognition. The parameter sensitivity of threshold θ used in the one-class leaf model was analyzed to validate the robustness of the MDCC. All the experiments were conducted on a 64-bit machine with Intel(R) Core(TM) quad-core processor (i7CPU@ 3.40GHz) and the NVIDIA GPU GTX1070.

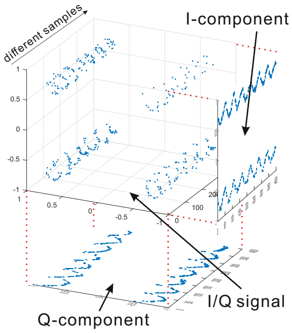


Figure 3: The I/Q data is illustrated as a 3-D graph; the 2-D graphs are two projections of the 3-D graph. When I/Q data are considered as two separate series of real numbers, the underlying dependency between them will fail to be utilized.

5.1 Experimental Setup

5.1.1 Datasets. Two real-world datasets were used to validate the performance of the proposed MDCC[1]. The process of data collection and the experiment settings utilized are detailed in this section.

RF signal Datasets. We collected RF signals to test the effectiveness of our method for real world applications. The objective of RF signal recognition is to recognize different RF devices based on the signals they transmitted. Each sample bag is a time-series signal which has complex values consisting of a real in-phase component (I) and an imaginary quadrature component (Q), as shown in Fig. 3. Each signal sample also comes with an identity label representing the unique device (e.g., transmitter) it originates from. For RF signal collection, we set up a wireless line-of-sight communication system with a static transmitter and receiver. The distance between the receiver and transmitter was fixed at 50cm. The same USRP-2943R device was utilized as the receiver and fix it through-out the entire testing process. In the receiver part, The receiver recieves the transmitted modulated signal and down converts it through the RF circuits before sending it on to the LabVIEW software for recording. We used two USRP-2943R and two USRP-N210 devices as transmitters in this experiment. This dataset was chosen originally due to the practical reasons and the difficulty of feature extraction tasks.

[1] The data and the code are available at: https://github.com/xguo7/MDCC-for-open-world-recognition

Eight sets of RF signal samples were collected and labeled from 1 to 8 in turn. The signal series for each class consisted of 4,096,000 data points for each of 4 columns, representing the values for the real and imaginary parts of both the received IQ sample and the associated standard. The data was split into 5,000 samples, with each sample consisting of 512 data points. Among the 5,000 samples of each class, 1,000 were collected in the buffer as the training set, with the indication that they are from the same long signal series, and the remainder used for testing.

Twitter dataset. The proposed MDCC was also validated for s Twitter topic recognition task. Here the data were collected using Twitter's Streaming API (filter/track method) to obtain a stream of public tweets filtered according to a given keyword set. Each collected tweet contains metadata along with the tweet message content, including the time of posting, location of source if available, and author profile information such as the author profile description, and number of followers and friends, number of statuses, across 11 days from 27 October 2012 to November 7 2012. Due to the limitations of Twitter's API policies, we utilized 4.9 million tweets borrowed from a labeled sample of tweets originally used in a prior study published in the crisis informatics literature [28], where the labeled classes included the following: Clothing, Food, Medical supplies including blood, Money, Shelter, Volunteer work. The labels were obtained using a crowd sourcing approach. Each collected tweet was then represented by an average word embedding vector using pretrained Twitter-GloVe embeddings [26] as inputs, each with a length of 200. 80 samples of each class were randomly chosen and assigned an identification to be collected in a buffer as training samples, while a further 80 samples of each class were used for testing.

5.1.2 Evaluation protocol. In the open world problem, the training phase is a complex process as new classes are continually being added to update the model. The open set evaluation protocol proposed by [2] needs certain unknown labels to validate several parameters and is thus not suitable for open-world problem. An online learning protocol by [23] does not consider the error propagation. We therefore propose a new protocol that can reflects real world scenario s more realistically.

Training Phase: For the training phase, training samples for each class were mixed together. Samples from two classes were deemed known for training the root node, with samples from new classes arriving one by one to be detected and learned. For example, in Stage 1, classes 1 and 2 are known, and class 3 is unknown. And we only test the stages where both known classes and unknown classes exist. The samples detected as new and placed in class 3 are then used to update the model.

Testing Phase: There are additional testing samples included for each class. The testing is conducted at every stage after the model has been updated. All the testing samples are used for every test. For example, in Stage 3 all the trained child models (initial model, one class model 1 and one class model 2) are tested by the combination of known samples and unseen samples.

To evaluate the performance, we utilize two measures, EN-Accuracy and F-measure [23]. EN-Accuracy is computed as:

$$EN - Accuracy = \frac{(N - known + N - unknown)}{N}, \tag{3}$$

where N is the total number of testing samples. N-known and N-unknown are the number of correctly recognized samples with known classes and with unknown classes, respectively. The F-score is defined as:

$$F - score = \frac{2TP}{(2TP + FP + FN)}, \tag{4}$$

where TP (True Positive) represents the correctly classified samples of unknown classes, FP (False Positive) is the number of incorrectly classified samples of unknown classes, and FN (False Negative) represents the number of incorrectly classified samples for the known classes.

5.1.3 Comparison algorithms. To validate the priority of the proposed method, several classical methods were also included here for comparison. A brief description of the methods is as follows:

- 1) Local novel detector (LOD) [4] is a multi-class novel detection method that can be combined with multi-class SVM [6] classifier to solve the open world problem.
- 2) R-Openmax: Openmax [3] was combined with a retraining policy for incremental learning.
- 3) S-Forest [23] is an online learning method ofen used to deal with streaming data. During the incremental process, it collects new samples and updates the model once the number of samples reaches a certain threshold.

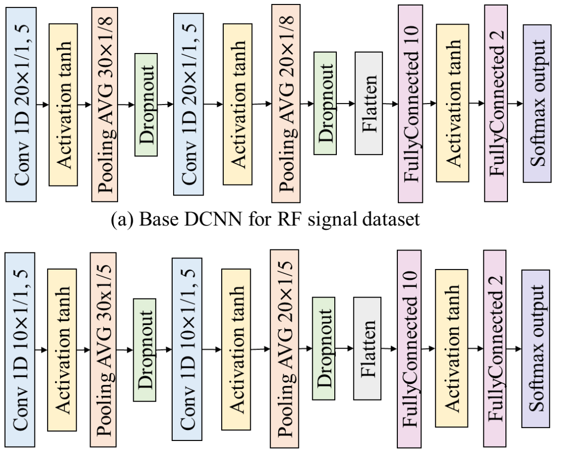
For methods like LOD and S-Forest, features of RF signals have to be manually extracted. 12 classical indicators of signals were therefore used as the features, namely the Mean, Median, Median Absolute Deviation, Standard Deviation, Skewness, Kurtosis, Max, Min, Mean Square, Root Mean Square, Pearson-Skewness, and Mean Absolute Deviation.

5.1.4 Architecture of the base DCNN model. Fig. 4 shows the architecture of the base DCNN model used as the root and leaf node in the proposed MDCC for the two real-world datasets. Both root and leaf nodes share the same architecture in the MDCC. The openmax layer of the root node is eliminated here since it is computed based on the output of the softmax layer. The structure of the convolution layer is expressed as: $< filter\ size/strides,\ number\ of\ filters >$. The structure of the pooling layer is expressed as: $<\ pooling\ size/strides >$. The dropout probability of each layer is 50%.

5.1.5 Parameters used in the experiments. Table 2 lists the parameters used in the training and testing phase in the experiments. lr-root and lr-leaf refer to the learning rate of optimization for the training root model and leaf model, respectively. α_{root} refers to the number of âĂIJtopâĂİ classes that are chosen for revision in order to compute the openmax output in the root model. γ_{root} refers to the rejection threshold of the openmax layer to define the new class of root model. The details of these two parameters can be found in [3]. The term θ refers to the estimated percentage of outlier instances when building the classifier for the leaf node, which determines the rejection line of the new classes.

5.2 Experiment Performance

5.2.1 Convergence of loss functions. There are two kinds of loss incurred in training a MDCC model: outer-describing loss and intra distance loss (i.e.self-describing regularization). To show the



(a) Base DCNN for RF signal dataset

(b) Base DCNN for twitter dataset

Figure 4: Architecture details of DCNN base model used in the two datasets.

Table 2: Parameters used in the experiments

Dataset	lr-root	lr-leaf	batch-size	α_{root}	γ_{root}	θ
RF signal	0.005	0.002	50	2	0.008	0.7
Twitter	0.001	0.002	20	2	0.008	0.5

convergence of the two kinds of loss and validate the effectiveness of the training strategy, we recorded the convergence process for the outer-describing loss, intra-distance loss and their total loss. Figure. 5 shows the loss convergence process for the root model and three newly added leaf models. The outer describing loss and intra-distance loss convergence synchronously and experience a steady decrease to around 0, after around 10,000 iterations. These results confirm that the MDCC with a self-describing regularization can be well trained to minimize both the outer-describing loss and intra-distance loss.

5.2.2 Performance evaluation on RF signal dataset. Table 3 shows the EN-Accuracy and F-score evaluated from Stage 1 to 6 by the proposed MDCC and the comparison methods. These results validate the effectiveness of MDCC, which achieved the best performance in 75% of the stages for both the EN-Accuracy and F-score; it also exhibited a stable performance as the number of classes increased. Specifically, MDCC outperformed both LOD and S-Forest by 29.1% and 23.2%, respectively, in EN-Accuracy, and 19% and 37.6%, respectively, in F-score. This because both S-Forest and LOD utilize handcrafted features defined initially, while MDCC updates and extracts dynamic features through supervised learning by a deep model. MDCC achived a performance that was similar to that of R-openmax during the first stage, but then went on to outperforms it by 13.4% in EN-Accuracy and 49.5% in F-score. This is because although R-openmax has a high feature extraction ability, the model tends to deteriorate quickly as new classes emerge since its current features are insufficient to learn more new classes. In addition, these results show that the cascade framework of the proposed MDCC contributes greatly to its overall performance.

Table 3: EN-Accuracy (EN-Acc) and F-score from Stage 1 to Stage 6 for the different methods tested on the RF signal dataset

Metric	Stage	LOD	S-Forest	R-openmax	MDCC
EN-Acc(%)	1	36.95	45.77	**60.45**	**60.45**
	2	34.85	33.24	43.08	**57.32**
	3	53.35	37.03	42.75	**54.50**
	4	33.83	55.20	**58.66**	50.43
	5	25.83	33.68	54.39	**55.65**
	6	31.87	32.38	33.55	**36.55**
F-score	1	**0.50**	0.33	0.41	0.41
	2	0.50	0.27	0.29	**0.75**
	3	0.60	0.19	0.17	**0.62**
	4	0.34	**0.62**	0.31	0.42
	5	0.18	0.11	0.27	**0.40**
	6	0.14	0.11	0.10	**0.19**

5.2.3 Performance evaluation on the Twitter dataset. Table 4 shows the performances in terms of EN-Accuracy and F-score from Stage 1 to 4 for the Twitter dataset achieved by the different methods. In general, MDCC achieved the best performance in 75% of the stages in both EN-Accuracy and F-score. Specifically, MDCC outperformed LOD and S-Forest by 18.5% and 6.25%, respevtively, for the EN-Accuracy, and 32% and 39.7%, respectively, for the F-score. This is because MDCC can update and extract the features through end-to-end supervised learning by deep neural networks for this difficult recognition task and takes into account the errors generated from the last stage. MDCC performed equally to R-openmax in the first stage, but then subsequently outperformed it by 13.4% in EN-Accuracy and 39.5% in F-score. This is because the predefined architecture of R-openmax limits the types and number of features used to distinguish incoming new classes.

Table 4: EN-Accuracy (EN-Acc) and F-score from Stage 1 to Stage 4 for the different methods tested on the Twitter dataset

Metric	Stage	LOD	S-Forest	R-openmax	MDCC
EN-Acc(%)	1	37.50	43.12	**49.53**	**49.53**
	2	18.33	35.00	40.70	**44.78**
	3	42.01	33.44	**42.02**	35.77
	4	31.50	34.27	34.50	**35.90**
F-score	1	0.27	0.31	**0.78**	**0.78**
	2	0.31	0.28	0.61	**0.62**
	3	0.39	0.25	**0.40**	0.37
	4	0.16	0.19	0.15	**0.22**

5.2.4 Sensitivity of the Classifier Threshold. There is one main parameter in the proposed MDCC, namely the percentage threshold θ in the leaf node used for filtering the outlier instances, which determines the rejection line of new classes. We used the same

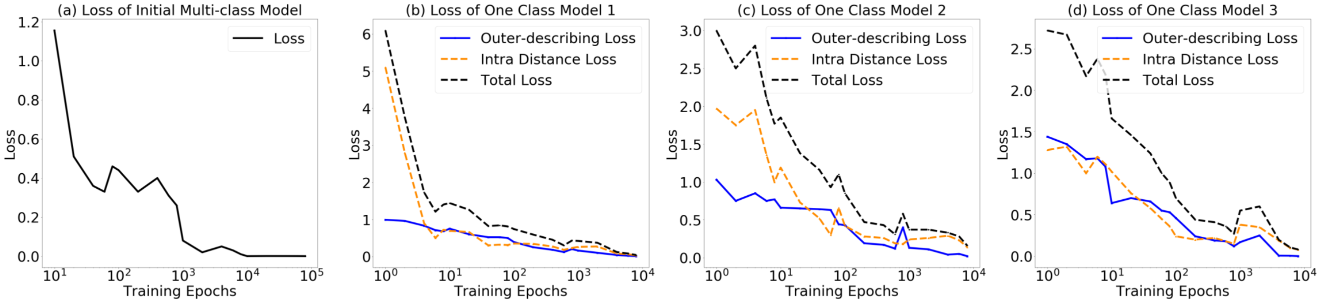


Figure 5: Convergence of loss functions

initial conditions and datasets as those mentioned before to test the parameter sensitivity. Fig. 6 shows the sensitivity results in Stage 3 of varying θ in two datasets. Stage 3 was chosen because the number of known and unknown classes at that stage are balanced. The threshold θ varies from 0.1 to 1. In general, for both datasets, the F-score and EN-Accuracy are slightly lower than for the other when $\theta > 0.5$, but when it is between 0.1 and 0.5, the result becomes stable. This is because the presumption that the outlier samples will be generated from the last node is over-estimated and more than half of the detected new samples are filtered. Overall, the threshold θ remains stable and thus achieves a good performance over a reasonable range between $0.1 \sim 0.5$.

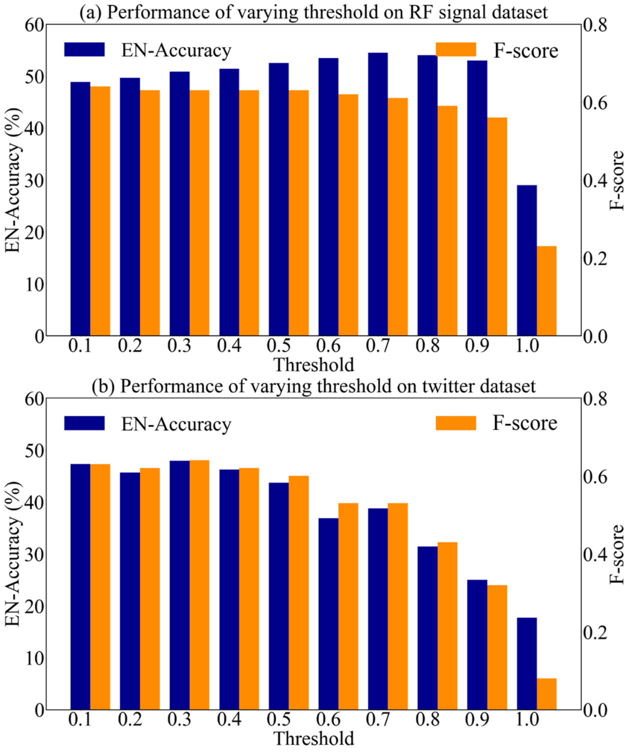


Figure 6: EN-Accuracy and F-score for different thresholds for (a) RF signal dataset and (b) Twitter dataset.

6 CONCLUSIONS

In this work, we addressed three challenges by proposing the end-to-end Multi-stage Deep Classifier Cascades for open world problem. First, the multi-class cascade architecture was implemented to incrementally detect and learn new coming classes. Second, an error-corrective one class classifier trained based on a dynamic reference set is proposed, where the leaf nodes learn unique features for each new coming class. Third, a feature space regularization based on the collective target set was utilzed to enforce the feature consistency of all instances in the same target set. Two real world experiments on the comparison to several existing methods indicate the effectiveness and efficiency of the proposed method. Tests to analyze the model parameter threshold indicated that a stable performance was achieved when the threshold was within a reasonable range.

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation grant: #1755850, #1841520, #1907805, Jeffress Trust Award, and NVIDIA GPU Grant.

REFERENCES

[1] Ferit Akova, Murat Dundar, Yuan Qi, and Bartek Rajwa. 2012. Self-adjusting models for semi-supervised learning in partially observed settings. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 21–30.

[2] Abhijit Bendale and Terrance Boult. 2015. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1893–1902.

[3] Abhijit Bendale and Terrance E Boult. 2016. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1563–1572.

[4] Paul Bodesheim, Alexander Freytag, Erik Rodner, and Joachim Denzler. 2015. Local novelty detection in multi-class recognition problems. In *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 813–820.

[5] Gert Cauwenberghs and Tomaso Poggio. 2001. Incremental and decremental support vector machine learning. In *Advances in neural information processing systems*. 409–415.

[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 27.

[7] Rocco De Rosa, Thomas Mensink, and Barbara Caputo. 2016. Online open world recognition. *arXiv preprint arXiv:1604.02275* (2016).

[8] Jia Deng, Sanjeev Satheesh, Alexander C Berg, and Fei Li. 2011. Fast and balanced: Efficient label tree learning for large scale object recognition. In *Advances in Neural Information Processing Systems*. 567–575.

[9] Murat Dundar, Ferit Akova, Alan Qi, and Bartek Rajwa. 2012. Bayesian nonexhaustive learning for online discovery and modeling of emerging classes. *arXiv preprint arXiv:1206.4600* (2012).

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[11] Victor Fragoso, Pradeep Sen, Sergio Rodriguez, and Matthew Turk. 2013. EVSAC: accelerating hypotheses generation by modeling matching scores with extreme value theory. In *Proceedings of the IEEE International Conference on Computer Vision*. 2472–2479.

[12] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211* (2013).

[13] Xiaojie Guo, Liang Chen, and Changqing Shen. 2016. Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis. *Measurement* 93 (2016), 490–502.

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

[15] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. 2006. Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research* 7, Sep (2006), 1909–1936.

[16] Qi Lei, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S Dhillon. 2017. Similarity preserving representation learning for time series analysis. *arXiv preprint arXiv:1702.03584* (2017).

[17] Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, Kai Yu, Liangliang Cao, and Thomas Huang. 2011. Large-scale image classification: fast feature extraction and svm training. (2011).

[18] Baoyuan Liu, Fereshteh Sadeghi, Marshall Tappen, Ohad Shamir, and Ce Liu. 2013. Probabilistic label trees for efficient large scale image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 843–850.

[19] Vincent Lonij, Ambrish Rawat, and Maria-Irina Nicolae. 2017. Open-World Visual Recognition Using Knowledge Graphs. *arXiv preprint arXiv:1708.08310* (2017).

[20] Markos Markou and Sameer Singh. 2003. Novelty detection: a reviewâĂŤpart 1: statistical approaches. *Signal processing* 83, 12 (2003), 2481–2497.

[21] Marcin Marszałek and Cordelia Schmid. 2008. Constructing category hierarchies for visual recognition. In *European conference on computer vision*. Springer, 479–491.

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

[23] Xin Mu, Kai Ming Ting, and Zhi-Hua Zhou. 2017. Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE Transactions on Knowledge and Data Engineering* 29, 8 (2017), 1605–1618.

[24] Miguel Nicolau, James McDermott, et al. 2016. A hybrid autoencoder and density estimation model for anomaly detection. In *International Conference on Parallel Problem Solving from Nature*. Springer, 717–726.

[25] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. 2015. Deep face recognition.. In *bmvc*, Vol. 1. 6.

[26] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[27] Pramuditha Perera and Vishal M Patel. 2018. Learning deep features for one-class classification. *arXiv preprint arXiv:1801.05365* (2018).

[28] Hemant Purohit, Carlos Castillo, Fernando Diaz, Amit Sheth, and Patrick Meier. 2014. Emergency-relief coordination on social media: Automatically matching resource requests and offers. *First Monday* 19, 1 (2014).

[29] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2001–2010.

[30] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016).

[31] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. 2013. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence* 35, 7 (2013), 1757–1772.

[32] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. 2014. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence* 36, 11 (2014), 2317–2324.

[33] Walter J Scheirer, Anderson Rocha, Ross J Micheals, and Terrance E Boult. 2011. Meta-recognition: The theory and practice of recognition score analysis. *IEEE transactions on pattern analysis and machine intelligence* 33, 8 (2011), 1689–1695.

[34] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[35] Julia S Thrower, Laura Hoffman, Martin Rechsteiner, and Cecile M Pickart. 2000. Recognition of the polyubiquitin proteolytic signal. *The EMBO journal* 19, 1 (2000), 94–102.

[36] Junxiang Wang, Yuyang Gao, Andreas Züfle, Jingyuan Yang, and Liang Zhao. 2018. Incomplete Label Uncertainty Estimation for Petition Victory Prediction with Dynamic Features. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 537–546.

[37] Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. 2018. Random warping series: A random features method for time-series embedding. *arXiv preprint arXiv:1809.05259* (2018).

[38] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. 2014. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 177–186.

[39] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2018. Learning to Accept New Classes without Training. *arXiv preprint arXiv:1809.06004* (2018).

[40] Tom Yeh and Trevor Darrell. 2008. Dynamic visual category learning. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

[41] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. 2005. A probabilistic model for online document clustering with application to novelty detection. In *Advances in neural information processing systems*. 1617–1624.

[42] Xuchao Zhang, Shuo Lei, Liang Zhao, Arnold Boedihardjo, and Chang-Tien Lu. 2018. Robust regression via online feature selection under adversarial data corruption. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1440–1445.

[43] Liang Zhao, Amir Alipour-Fanid, Martin Slawski, and Kai Zeng. 2018. Prediction-time Efficient Classification Using Feature Computational Dependencies. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2787–2796.

[44] Liang Zhao, Junxiang Wang, and Xiaojie Guo. 2018. Distant-supervision of heterogeneous multitask learning for social event forecasting with multilingual indicators. In *Thirty-Second AAAI Conference on Artificial Intelligence*.