Learning Deep Features for One-Class Classification

Pramuditha Perera, *Student Member, IEEE,* and Vishal M. Patel, *Senior Member , IEEE*

Abstract—We propose a deep learning-based solution for the problem of feature learning in one-class classification. The proposed method operates on top of a Convolutional Neural Network (CNN) of choice and produces descriptive features while maintaining a low intra-class variance in the feature space for the given class. For this purpose two loss functions, *compactness loss* and *descriptiveness loss* are proposed along with a parallel CNN architecture. A template matching-based framework is introduced to facilitate the testing process. Extensive experiments on publicly available anomaly detection, novelty detection and mobile active authentication datasets show that the proposed Deep One-Class (DOC) classification method achieves significant improvements over the state-of-the-art.

Index Terms—One-class classification, anomaly detection, novelty detection, deep learning.

✦

1 INTRODUCTION

Fully supervised image classification is one of the most commonly studied problems in computer vision. Given a set of image-label pairs of different object classes, the goal of image classification[1] is to predict the class label of a test image seen during testing. Based on the number of classes present at training/validation and testing, image classification can be divided in to four sub-problems (multi-class classification, open-set classification, detection and one-class classification), as shown in Table 1. Majority of the past work have attempted to address the multi-class classification [21], [16], [8] and detection problems [17], [39], [12]. While open-set classification [32], [2], [40] has received some attention in recent years, little or no work has been done specifically targeting one-class image classification. In this work, we address the latter by engineering deep features targeting one-class classification tasks.

The objective of one-class classification is to recognize instances of a concept by only using examples of the same concept [15] as shown in Figure 1. In such a training scenario, instances of only a single object class are available during training. In the context of this paper, all other classes except the class given for training is called alien classes [2]. During testing, the classifier may encounter objects from alien classes. The goal of the classifier is to distinguish objects of the known class from objects of alien classes. It should be noted that one-class classification is different from binary classification due to the absence of training data from a second class.

One-class classification is encountered in many real-world computer vision applications including novelty detection [23], anomaly detection [5], [27], medical imaging and mobile active authentication [25]. In all of these applications, unavailability of samples from alien classes is either due to the openness of the
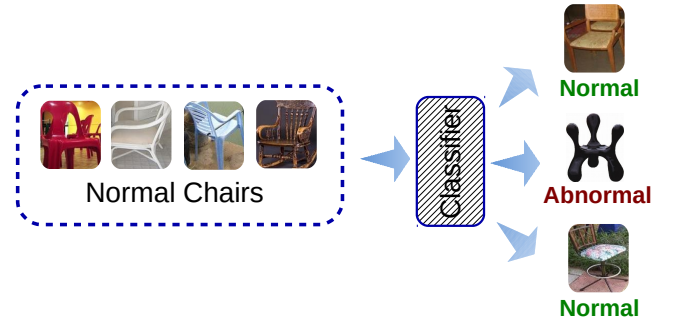

Fig. 1. In one class classification, given samples of a single class, a classifier is required to learn so that it can identify out-of-class(alien) objects. Abnormal image detection is an application of one class classification. Here, given a set of normal chair objects, a classifier is learned to detect abnormal chair objects.

TABLE 1
Number of classes available during training, validation and testing phases for each sub-classification task. Here, $2 \le M < N$.

	Training	Validation	Testing
Multi-class classification	N	N	N
Open-set classification	M	M	N
Binary classification (Detection)	2	2	2
One-class classification	1	1	N

problem or due to the high cost associated with obtaining the samples of such classes. For example, in a novelty detection application, it is counter intuitive to come up with novel samples to train a classifier. On the other hand, in mobile active authentication, samples of alternative classes (users) are often difficult to obtain due to the privacy concerns [22].

In order to solve the one-class classification problem, we seek motivation from generic object classification frameworks. Many previous works in object classification have focused on improving either the feature or the classifier (or in some cases both) in an attempt to improve classification performance. In particular, various deep learning-based feature extraction and classification methods have been proposed in the literature and have gained a lot of

---

- *P. Perera and V. M. Patel are with the Department of Electrical and Computer Engineering at Rutgers University, Piscataway, NJ 08854.*
  E-mail: {pramuditha.perera, vishal.m.patel}@rutgers.edu

1. Throughout the paper what we mean by image classification is fully supervised image classification.

2. Depending on the domain of application an alien class may be called by an alternative term. For example, an alien class is known as novel class, abnormal class/outlier class, attacker/intruder class in novelty detection, abnormal image detection and active-authentication applications, respectively.
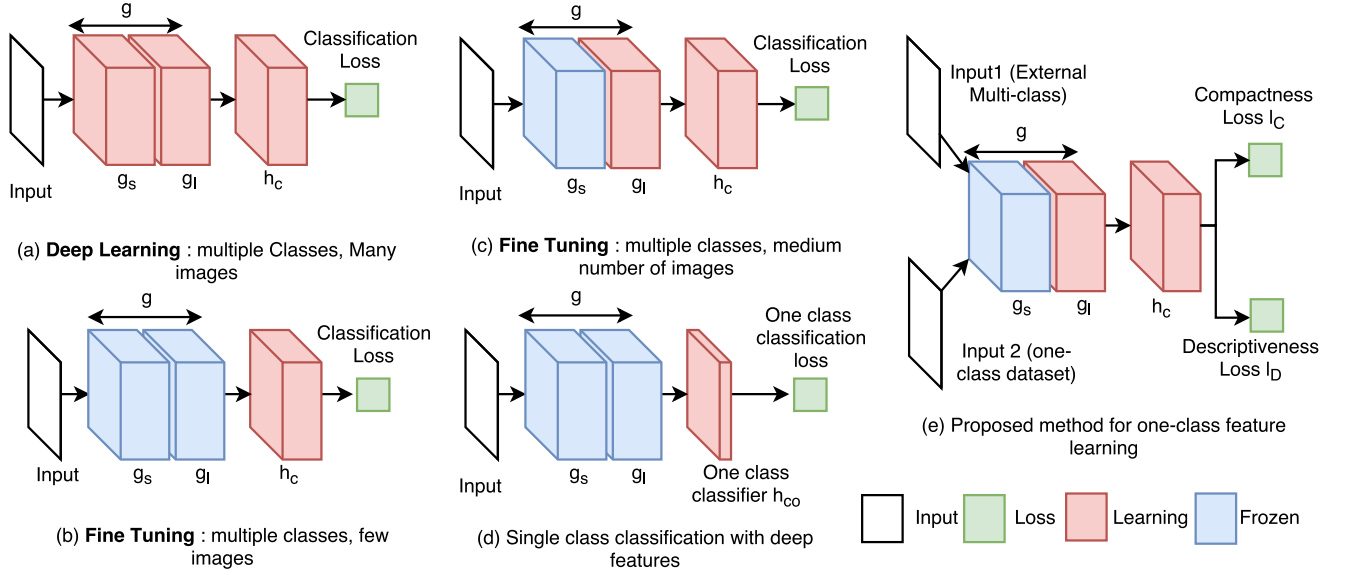
Fig. 2. Different deep learning strategies used for classification. In all cases, learning starts from a pre-trained model. Certain subnetworks (in blue) are frozen while others (in red) are learned during training. Setups (a)-(c) cannot be used directly for the problem of one-class classification. Proposed method shown in **(e)** accepts two inputs at a time - one from the given one-class dataset and the other from an external multi-class dataset and produces two losses.

traction in recent years [21], [34]. In general, deep learning-based classification schemes have two subnetworks, a feature extraction network (g) followed by a classification sub network (h_c), that are learned jointly during training. For example, in the popular Alexnet architecture [21], the collection of convolution layers may be regarded as (g) where as fully connected layers may collectively be regarded as (h_c). Depending on the output of the classification sub network (h_c), one or more losses are evaluated to facilitate training. Deep learning requires the availability of multiple classes for training and extremely large number of training samples (in the order of thousands or millions). However, in learning tasks where either of these conditions are not met, the following alternative strategies are used:

(a) Multiple classes, many training samples: This is the case where both requirements are satisfied. Both feature extraction and classification networks, g and h_c are trained end-to-end (Figure 2(a)). The network parameters are initialized using random weights. Resultant model is used as the pre-trained model for fine tuning [21], [16].

(b) Multiple classes, low to medium number of training samples: The feature extraction network from a pre-trained model is used. Only a new classification network is trained in the case of low training samples (Figure 2(b)). When medium number of training samples are available, feature extraction network (g) is divided into two sub-networks - shared feature network (g_s) and learned feature network (g_l), where $g = g_s \circ g_l$. Here, g_s is taken from a pre-trained model. g_l and the classifier are learned from the data in an end-to-end fashion (Figure 2(c)). This strategy is often referred to as fine-tuning [12].

(c) Single class or no training data: A pre-trained model is used to extract features. The pre-trained model used here could be a model trained from scratch (as in (a)) or a model resulting from fine-tuning (as in (b)) [9], [22] where training/fine-tuning is performed based on an external dataset. When training data from a class is available, a one-class classifier is trained on the extracted

features (Figure 2(d)).

In this work, we focus on the task presented in case (c) where training data from a single class is available. Strategy used in case (c) above uses deep-features extracted from a pre-trained model, where training is carried out on a different dataset, to perform one-class classification. However, there is no guarantee that features extracted in this fashion will be as effective in the new one-class classification task. In this work, we present a feature fine tuning framework which produces deep features that are specialized to the task at hand. Once the features are extracted, they can be used to perform classification using the strategy discussed in (c).

In our formulation (shown in Figure 2 (e)), starting from a pre-trained deep model, we freeze initial features (g_s) and learn (g_l) and (h_c). Based on the output of the classification sub-network (h_c), two losses *compactness loss* and *descriptiveness loss* are evaluated. These two losses, introduced in the subsequent sections, are used to assess the quality of the learned deep feature. We use the provided one-class dataset to calculate the *compactness loss*. An external multi-class reference dataset is used to evaluate the *descriptiveness loss*. As shown in Figure 3, weights of g_l and h_c are learned in the proposed method through back-propagation from the composite loss. Once training is converged, system shown in setup in Figure 2(d) is used to perform classification where the resulting model is used as the pre-trained model.

In summary, this paper makes the following three contributions.

1) We propose a deep-learning based feature engineering scheme called Deep One-class Classification (DOC), targeting one-class problems. To the best of our knowledge this is the first work to address this problem.

2) We introduce a joint optimization framework based on two loss functions - *compactness loss* and *descriptiveness loss*. We propose *compactness loss* to assess the compactness of the class under consideration in the learned feature space. We propose using an external multi-class
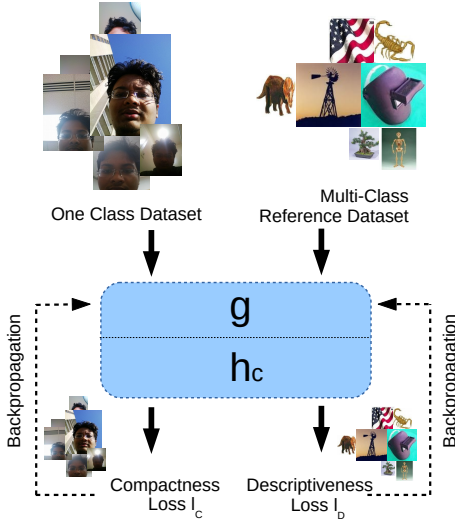
Fig. 3. Overview of the proposed method. In addition to the provided one-class dataset, another arbitrary multi-class dataset is used to train the CNN. The objective is to produce a CNN which generates descriptive features which are also compactly localized for the one-class training set. Two losses are evaluated independently with respect to the two datasets and back-propagation is performed to facilitate training.

dataset to assess the descriptiveness of the learned features using *descriptiveness loss*.

3) On three publicly available datasets, we achieve state-of-the-art one-class classification performance across three different tasks.

Rest of the paper is organized as follows. In Section 2, we review a few related works. Details of the proposed deep one-class classification method are given in Section 3. Experimental results are presented in Section 4. Section 5 concludes the paper with a brief discussion and summary.

2 RELATED WORK

Generic one-class classification problem has been addressed using several different approaches in the literature. Most of these approaches predominately focus on proposing a better classification scheme operating on a chosen feature and are not specifically designed for the task of one-class image classification. One of the initial approaches to one-class learning was to estimate a parametric generative model based on the training data. Work in [26], [3], propose to use Gaussian distributions and Gaussian Mixture Models (GMMs) to represent the underlying distribution of the data. In [19] comparatively better performances have been obtained by estimating the conditional density of the one-class distribution using Gaussian priors.

The concept of Support Vector Machines (SVMs) was extended for the problem of one-class classification in [33]. Conceptually this algorithm treats the origin as the out-of-class region and tries to construct a hyperplane separating the origin with the class data. Using a similar motivation, [35] proposed Support Vector Data Description (SVDD) algorithm which isolates the training data by constructing a spherical separation plane. Practical one-class learning applications on different domains are predominantly developed based on these conceptual bases.

In [29], visual anomalies in wire ropes are detected based on Gaussian process modeling. Anomaly detection is performed by

maximizing KL divergence in [28], where the underlying distribution is assumed to be a known Gaussian. A detailed description of various anomaly detection methods can be found in [5].

Novelty detection based on one-class learning has also received a significant attention in recent years. Some of the earlier works in novelty detection focused on estimating a parametric model for data and to model the tail of the distribution to improve classification accuracy [7], [27]. In [4], null space-based novelty detection framework for scenarios when a single and multiple classes are present is proposed. However, it is mentioned in [4] that their method does not yield superior results compared with the classical one-class classification methods when only a single class is present. An alternative null space-based approach based on kernel Fisher discriminant was proposed in [10] specifically targeting one-class novelty detection. A detailed survey of different novelty detection schemes can be found in [23], [24].

Mobile-based Active Authentication (AA) is another application of one-class learning which has gained interest of the research community in recent years [25]. In mobile AA, the objective is to continuously monitor the identity of the user based on his/her enrolled data. As a result, only the enrolled data (i.e. one-class data) are available during training. Some of the recent works in AA has taken advantage of CNNs for classification. Work in [31], uses a CNN to extract attributes from face images extracted from the mobile camera to determine the identity of the user. Various deep feature-based AA methods have also been proposed as benchmarks in [22] for performance comparison.

Since one-class learning is constrained with training data from only a single class, it is impossible to adopt a CNN architectures used for classification [21], [34] and verification [6] directly for this problem. In the absence of a discriminative feature generation method, in most unsupervised tasks, the activation of a deep layer is used as the feature for classification. This approach is seen to generate satisfactory results in most applications [9]. This can be used as a starting point for one-class classification as well. As an alternative, autoencoders [37], [14] and variants of autoencoders [38], [20] can also to be used as feature extractors for one-class learning problems. However, in this approach, knowledge about the outside world is not taken into account during the representation learning. Furthermore, none of these approaches were specifically designed for the purpose of one-class learning. To the best of our knowledge, one-class feature learning has not been addressed using a deep-CNN architecture to date.

3 DEEP ONE-CLASS CLASSIFICATION (DOC)
3.1 Overview

In this section, we formulate the objective of one-class feature learning as an optimization problem. In the classical multiple-class classification, features are learned with the objective of maximizing inter-class distances between classes and minimizing intra-class variances within classes [1]. However, in the absence of multiple classes such a discriminative approach is not possible.

In this light, we outline two important characteristics of features intended for one-class classification.

Compactness $\mathcal{C}$. A desired quality of a feature is to have a similar feature representation for different images of the same class. Hence, a collection of features extracted from a set of images of a given class will be compactly placed in the feature space. This quality is desired even in features used for multi-class classification. In such cases, compactness is quantified using the

intra-class distance [1]; a compact representation would have a lower intra-class distance.

Descriptiveness $\mathcal{D}$**.** The given feature should produce distinct representations for images of different classes. Ideally, each class will have a distinct feature representation from each other. Descriptiveness in the feature is also a desired quality in multi-class classification. There, a descriptive feature would have large inter-class distance [1].

It should be noted that for a useful (discriminative) feature, both of these characteristics should be satisfied collectively. Unless mutually satisfied, neither of the above criteria would result in a useful feature. With this requirement in hand, we aim to find a feature representation g that maximizes both compactness and descriptiveness. Formally, this can be stated as an optimization problem as follows,

$$\hat{g} = \max_g \ \mathcal{D}(g(t)) + \lambda \mathcal{C}(g(t)), \tag{1}$$

where t is the training data corresponding to the given class and λ is a positive constant. Given this formulation, we identify three potential strategies that may be employed when deep learning is used for one-class problems. However, none of these strategies collectively satisfy both descriptiveness and compactness.

(a) Extracting deep features. Deep features are first extracted from a pre-trained deep model for given training images. Classification is done using a one-class classification method such as one-class SVM, SVDD or k-nearest neighbor using extracted features. This approach does not directly address the two characteristics of one-class features. However, if the pre-trained model used to extract deep features was trained on a dataset with large number of classes, then resulting deep features are likely to be descriptive. Nevertheless, there is no guarantee that the used deep feature will possess the compactness property.

(b) Fine-tune a two class classifier using an external dataset. Pre-trained deep networks are trained based on some legacy dataset. For example, models used for the ImageNet challenge are trained based on the ImageNet dataset [8]. It is possible to fine tune the model by representing the alien classes using the legacy dataset. This strategy will only work when there is a high correlation between alien classes and the legacy dataset. Otherwise, the learned feature will not have the capacity to describe the difference between a given class and the alien class thereby violating the descriptiveness property.

(c) Fine-tune using a single class data. Fine-tuning may be attempted by using data only from the given single class. For this purpose, minimization of the traditional cross-entropy loss or any other appropriate distance could be used. However, in such a scenario, the network may end up learning a trivial solution due to the absence of a penalty for miss-classification. In this case, the learned representation will be compact but will not be descriptive.

Let us investigate the appropriateness of these three strategies by conducting a case study on the abnormal image detection problem where the considered class is the *normal chair* class. In abnormal image detection, initially a set of *normal chair* images are provided for training as shown in Figure 4(a). The goal is to learn a representation such that, it is possible to distinguish a *normal chair* from an *abnormal chair*.

The trivial approach to this problem is to extract deep features from an existing CNN architecture (solution (a)). Let us assume that the Alexnet architecture [21] is used for this purpose and *fc7* features are extracted from each sample. Since deep features are sufficiently descriptive, it is reasonable to expect samples of the same class to be clustered together in the extracted feature space. Illustrated in Figure 4(b) is a 2D visualization of the extracted 4096 dimensional features using t-SNE [36]. As can be seen from Figure4(b), the Alexnet features are not able to enforce sufficient separation between normal and abnormal chair classes.

Another possibility is to train a two class classifier using the Alexnet architecture by providing normal chair object images and the images from the ImageNet dataset as the two classes (solution (b)). However, features learned in this fashion produce similar representations for both normal and abnormal images, as shown in Figure4(c). Even though there exist subtle differences between normal and abnormal chair images, they have more similarities compared to the other ImageNet objects/images. This is the main reason why both normal and abnormal images end up learning similar feature representations.

A naive, and ineffective, approach would be to fine-tune the pre-trained Alexnet network using only the *normal chair* class (solution (c)). Doing so, in theory, should result in a representation where all *normal chairs* are compactly localized in the feature space. However, since all class labels would be identical in such a scenario, the fine-tuning process would end up learning a futile representation as shown in Figure4(d). The reason why this approach ends up yielding a trivial solution is due to the absence of a regularizing term in the loss function that takes into account the discriminative ability of the network. For example, since all class labels are identical, a zero loss can be obtained by making all weights equal to zero. It is true that this is a valid solution in the closed world where only *normal chair* objects exist. But such a network has zero discriminative ability when *abnormal chair* objects appear.

None of the three strategies discussed above are able to produce features that are both compact and descriptive. We note that out of the three strategies, the first produces the most reasonable representation for one-class classification. However, this representation was learned without making an attempt to increase compactness of the learned feature. Therefore, we argue that if compactness is taken into account along with descriptiveness, it is possible to learn a more effective representation.

3.2 Proposed Loss Functions

In this work, we propose to quantify compactness and descriptiveness in terms of measurable loss functions. Variance of a distribution has been widely used in the literature as a measure of the distribution spread [13]. Since spread of the distribution is inversely proportional to the compactness of the distribution, it is a natural choice to use variance of the distribution to quantify compactness. In our work, we approximate variance of the feature distribution by the variance of each feature batch. We term this quantity as the *compactness loss* (l_C).

On the other hand, descriptiveness of the learned feature cannot be assessed using a single class training data. However, if there exists a reference dataset with multiple classes, even with random object classes unrelated to the problem at hand, it can be used to assess the descriptiveness of the engineered feature. In other words, if the learned feature is able to perform classification with high accuracy on a different task, the descriptiveness of the learned feature is high. Based on this rationale, we use the learned feature to perform classification on an external multi-class dataset, and consider classification loss there as an indicator of the
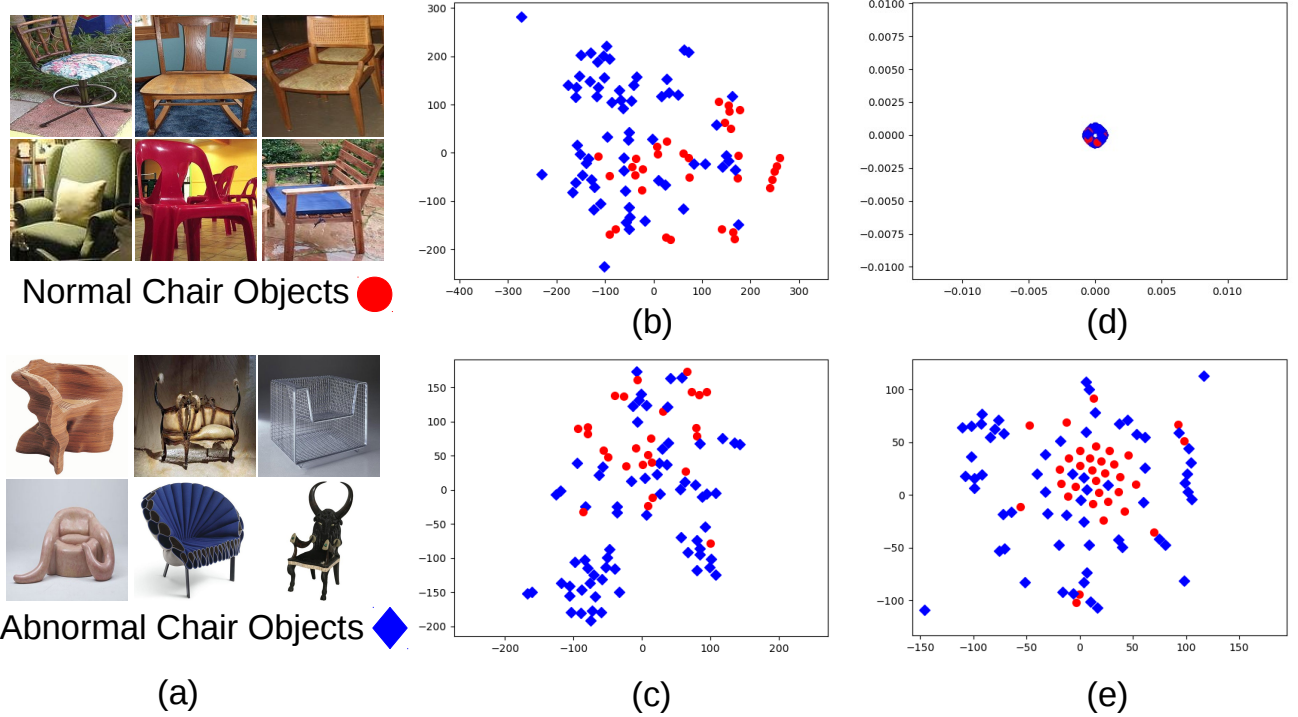
Fig. 4. Possible strategies for one-class classification in abnormal image detection. (a) Normal and abnormal image samples. (b) Feature space obtained using the Alexnet features. (c) Feature space obtained training a two class CNN with the alien objects represented by the ImageNet data samples. Normal and abnormal samples are not sufficiently separated in (b) and (c). (d) Feature space (2D Projection) obtained by fine-tuning by just using normal objects. Both normal and abnormal samples are projected to the same point. (e) Feature space (2D Projection) obtained using the proposed method. Normal and abnormal samples are relatively more separated. In all cases, feature space is visualized using tSNE.

descriptiveness of the learned feature. We call the cross-entropy loss calculated in this fashion as the *descriptiveness loss* (l_D). Here, we note that *descriptiveness loss* is low for a descriptive representation.

With this formulation, the original optimization objective in equation (1) can be re-formulated as,

$$\hat{g} = \min_g \ l_D(r) + \lambda l_C(t), \qquad (2)$$

where l_C and l_D are *compactness loss* and *descriptiveness loss*, respectively and r is the training data corresponding to the reference dataset. The tSNE visualization of the features learned in this manner for normal and abnormal images are shown in Figure 2(e). Qualitatively, features learned by the proposed method facilitates better distinction between normal and abnormal images as compared with the cases shown in Figure 2(b)-(d).

3.3 Terminology

Based on the intuition given in the previous section, the architecture shown in Figure 5 (a) is proposed for one-class classification training and the setup shown in Figure 5 (b) for testing. They consist of following elements:

Reference Network (R)**:** This is a pre-trained network architecture considered for the application. Typically it contains a repetition of convolutional, normalization, and pooling layers (possibly with skip connections) and is terminated by an optional set of fully connected layers. For example, this could be the Alexnet network [21] pre-trained using the ImageNet [8] dataset. Reference network can be seen as the composition of a feature

extraction sub-network g and a classification sub-network h_c. For example, in Alexnet, *conv1-fc7* layers can be associated with g and fc8 layer with h_c. *Descriptive loss* (l_D) is calculated based on the output of h_c.

Reference Dataset (r)**:** This is the dataset (or a subset of it) used to train the network R. Based on the example given, reference dataset is the ImageNet dataset [8] (or just a subset of the ImageNet dataset).

Secondary Network (S)**:** This is a second CNN where the network architecture is structurally identical to the reference network. Note that g and h_c are common in both of these networks. *Compactness loss* (l_C) is evaluated based on the output of h_c. For the considered example, S would have the same structure as R (Alexnet) up to *fc8*.

Target Dataset (t)**:** This dataset contains samples of the class for which one-class learning is used for. For example, for an abnormal image detection application, this dataset will contain normal images (i.e. data samples of the single class considered).

Model (W): This corresponds to the collection of weights and biases in the network, g and h_c. Initially, it is initialized by some pre-trained parameters W_0. [3]

Compactness loss (l_C) **:** All the data used during the training phase will belong to the same class. Therefore they all share the same class label. *Compactness loss* evaluates the average similarity between the constituent samples of a given batch. For a large enough batch, this quantity can be expected to represent average intra-class variance of a given class. It is desirable to

---

3. For the case of Alexnet, pre-trained model can be found at www.berkleyvison.com.

select a smooth differentiable function as l_C to facilitate back propagation. In our work, we define compactness loss based on the Euclidean distance.

Descriptiveness loss (l_D) **:** Descriptiveness loss evaluates the capacity of the learned feature to describe different concepts. We propose to quantify discriminative loss by the evaluation of cross-entropy with respect to the reference dataset (R).

For this discussion, we considered the Alexnet CNN architecture as the reference network. However, the discussed principles and procedures would also apply to any other CNN of choice. In what follows, we present the implementation details of the proposed method.

3.4 Architecture

The proposed training architecture is shown in Figure 5 (a) [4]. The architecture consists of two CNNs, the reference network (R) and the secondary network (S) as introduced in the previous subsection. Here, weights of the reference network and secondary network are tied across each corresponding counterparts. For example, weights between conv$i$ (where, $i = 1, 2.., 5$) layer of the two networks are tied together forcing them to be identical. All components, except *Compactness loss*, are standard CNN components. We denote the common feature extraction sub-architecture by $g(.)$ and the common classification by sub-architecture by $h_c(.)$. Please refer to Appendix for more details on the architectures of the proposed method based on the Alexnet and VGG16 networks.

3.5 Compactness loss

Compactness loss computes the mean squared intra-batch distance within a given batch. In principle, it is possible to select any distance measure for this purpose. In our work, we design compactness loss based on the Euclidean distance measure. Define $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_1}, \ldots, \mathbf{x_n}\} \in R^{n \times k}$ to be the input to the loss function, where the batch size of the input is n.

Forward Pass: For each ith sample $\mathbf{x_i} \in \mathbb{R}^k$, where $1 \leq i \leq n$, the distance between the given sample and the rest of the samples $\mathbf{z_i}$ can be defined as,

$$\mathbf{z_i} = \mathbf{x_i} - \mathbf{m_i}, \tag{3}$$

where, $\mathbf{m_i} = \frac{1}{n-1} \sum_{j \neq i} \mathbf{x_j}$ is the mean of rest of the samples. Then, compactness loss l_C is defined as the average Euclidean distance as in,

$$l_C = \frac{1}{nk} \sum_{i=1}^{n} \mathbf{z_i}^T \mathbf{z_i}. \tag{4}$$

Backpropagation:. In order to perform back-propagation using this loss, it is necessary to assess the contribution each element of the input has on the final loss. Let $\mathbf{x_i} = \{x_{i1}, x_{i2}, \ldots, x_{ik}\}$. Similarly, let $\mathbf{m_i} = \{m_{i1}, m_{i2}, \ldots, m_{ik}\}$. Then, the gradient of l_b with respect to the input x_{ij} is given as,

$$\frac{\partial l_C}{\partial x_{ij}} = \frac{2}{(n-1)nk} \left[n \times (x_{ij} - m_{ij}) - \sum_{k=1}^{n} (x_{ik} - m_{ik}) \right]. \tag{5}$$

4. Source code of the proposed method is made available online at https://github.com/PramuPerera/DeepOneClass

Detailed derivation of the back-propagation formula can be found in the Appendix. The loss l_C calculated in this form is equal to the sample feature variance of the batch multiplied by a constant (see Appendix). Therefore, it is an inverse measure of the compactness of the feature distribution.

3.6 Training

During the training phase, initially, both the reference network (R) and the secondary network (S) are initialized with the pre-trained model weights $\mathbf{W_0}$. Recall that except for the types of loss functions associated with the output, both these networks are identical in structure. Therefore, it is possible to initialize both networks with identical weights. During training, all layers of the network except for the last four layers are frozen as commonly done in network fine-tuning. In addition, the learning rate of the training process is set at a relatively lower value (5×10^{-5} is used in experiments). During training, two image batches, each from the reference dataset and the target dataset are simultaneously fed into the input layers of the reference network and secondary network, respectively. At the end of the forward pass, the reference network generates a *descriptiveness loss* (l_D), which is same as the cross-entropy loss by definition, and the secondary network generates *compactness loss* (l_C). The composite loss (l) of the network is defined as,

$$l(r, t) = l_D(r|W) + \lambda l_C(t|W), \tag{6}$$

where λ is a constant. It should be noted that, minimizing this loss function leads to the minimization of the optimization objective in (2).

In our experiments, λ is set equal to 0.1 Based on the composite loss, the network is back-propagated and the network parameters are learned using gradient descent or a suitable variant. Training is carried out until composite loss $l(r, t)$ converges. A sample variation of training loss is shown in Figure 6. In general, it was observed that composite loss converged in around two epochs (here, epochs are defined based on the size of the target dataset).

Intuitively, the two terms of the loss function l_D and l_C measure two aspects of features that are useful for one-class learning. Cross entropy loss values obtained in calculating *descriptiveness loss* l_D measures the ability of the learned feature to describe different concepts with respect to the reference dataset. Having reasonably good performance in the reference dataset implies that the learned features are discriminative in that domain. Therefore, they are likely to be descriptive in general. On the other hand, *compactness loss* (l_C) measures how compact the class under consideration is in the learned feature space. The weight λ governs the mutual importance placed on each requirement.

If λ is made large, it implies that the descriptiveness of the feature is not as important as the compactness. However, this is not a recommended policy for one-class learning as doing so would result in trivial features where the overlap between the given class and an alien class is significantly high. As an extreme case, if $\lambda = 0$ (this is equivalent to removing the reference network and carrying out training solely on the secondary network (Figure 4 (d)), the network will learn a trivial solution. In our experiments, we found that in this case the weights of the learned filters become zero thereby making output of any input equal to zero.

Therefore, for practical one-class learning problems, both reference and secondary networks should be present and more prominence should be given to the loss of the reference network.
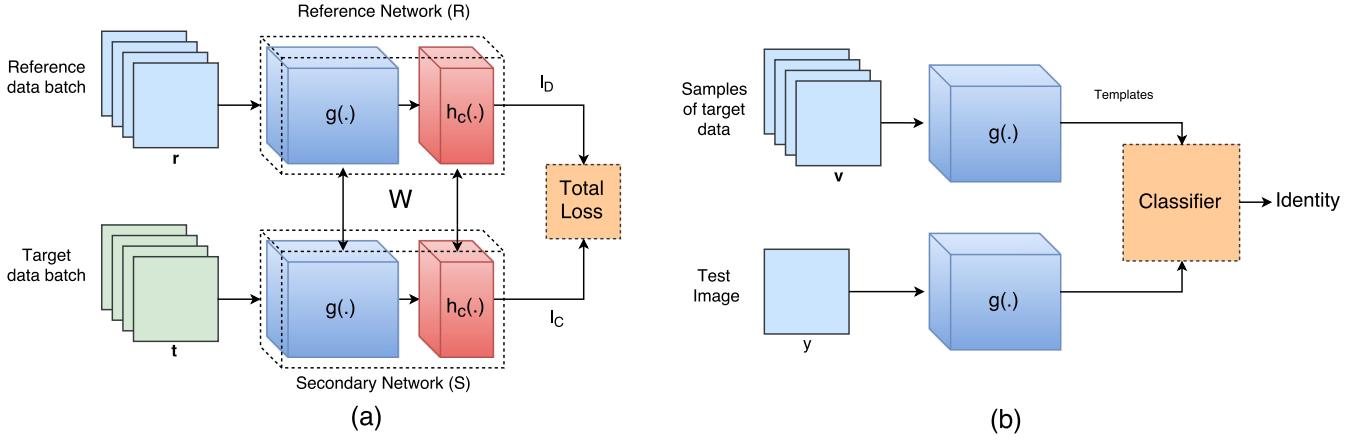
Fig. 5. (a) Training, and (b) testing frameworks of the proposed DOC method. During training, data from target and reference datasets are fed into the network simultaneously and training is done based on two losses, *descriptive loss* and *compactness loss*. During testing, only the feature extraction sub-network g is considered. A set of templates is first constructed by extracting features of a subset of the training dataset. When a test image is present, features of the test image is compared against templates using a classification rule to classify the test image.
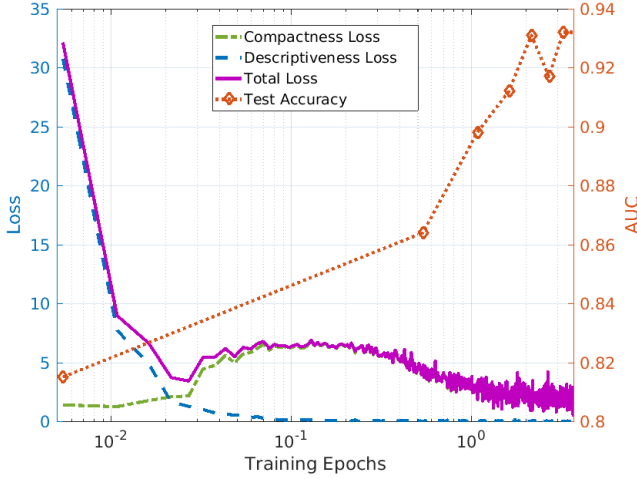

Fig. 6. Variation of loss functions during training in the proposed method obtained for the chair class in the abnormal image detection task. Training epochs are shown in the log scale. Compactness loss reaches stability around 0.1 epochs. Total loss (composite loss) reaches stability around 2 epochs. Highest test accuracy is obtained once composite loss converges.

3.7 Testing

The proposed testing procedure involves two phases - template generation and matching. For both phases, secondary network with weights learned during training is used as shown in Figure 5 (b). During both phases, the excitation map of the feature extraction sub-network is used as the feature. For example, layer 7, *fc7* can be considered from a Alexnet-based network. First, during the template generation phase a small set of samples $v = \{v_1, v_2, \ldots, v_n\}$ are drawn from the target (i.e. training) dataset where $v \in t$. Then, based on the drawn samples a set of features $g(v_1), g(v_2), \ldots, g(v_n)$ are extracted. These extracted features are stored as templates and will be used in the matching phase.

Based on stored template, a suitable one-class classifier, such as one-class SVM [33], SVDD [35] or k-nearest neighbor, can be

trained on the templates. In this work, we choose the simple k-nearest neighbor classifier described below. When a test image y is present, the corresponding deep feature $g(y)$ is extracted using the described method. Here, given a set of templates, a matched score S_y is assigned to y as

$$S_y = f(g(y)|g(t_1), g(t_2), \ldots, g(t_n)), \tag{7}$$

where $f(.)$ is a matching function. This matching function can be a simple rule such as the cosine distance or it could be a more complicated function such as Mahalanobis distance. In our experiments, we used Euclidean distance as the matching function. After evaluating the matched score, y can be classified as follows,

$$class(y) = \begin{cases} 1, & \text{if } S_y \leq \delta \\ 0, & \text{if } S_y > \delta, \end{cases} \tag{8}$$

where 1 is the class identity of the class under consideration and 0 is the identity of other classes and δ is a threshold.

3.8 Memory Efficient Implementation

Due to shared weights between the reference network and the secondary network, the amount of memory required to store the network is nearly twice as the number of parameters. It is not possible to take advantage of this fact with deep frameworks with static network architectures (such as caffe [18]). However, when frameworks that support dynamic network structures are used (e.g. PyTorch), implementation can be altered to reduce memory consumption of the network.

In the alternative implementation, only a single core network with functions g and h_c is used. Two loss functions l_C and l_D branch out from the core network. However in this setup, *descriptiveness loss* (l_D) is scaled by a factor of $1 - \lambda$. In this formulation, first λ is made equal to 0 and a data batch from the reference dataset is fed into the network. Corresponding loss is calculated and resulting gradients are calculated using back-propagation Then, λ is made equal to 1 and a data batch from the target dataset is fed into the network. Gradients are recorded same as before after back-propagation Finally, the average gradient is calculated using two recorded gradient values, and network

parameters are updated accordingly. In principle, despite of having a lower memory requirement, learning behavior in the alternative implementation would be identical to the original formulation.

4 EXPERIMENTAL RESULTS

In order to asses the effectiveness of the proposed method, we consider three one-class classification tasks: abnormal image detection, single class image novelty detection and active authentication. We evaluate the performance of the proposed method in all three cases against state of the art methods using publicly available datasets. Further, we provide two additional CNN-based baseline comparisons.

4.1 Experimental Setup

Unless otherwise specified, we used 50% of the data for training and the remaining data samples for testing. In all cases, 40 samples were taken at random from the training set to generate templates. In datasets with multiple classes, testing was done by treating one class at a time as the positive class. Objects of all the other classes were considered to be alien. During testing, alien object set was randomly sampled to arrive at equal number of samples as the positive class. As for the reference dataset, we used the validation set of the ImageNet dataset for all tests. When there was an object class overlap between the target dataset and the reference dataset, the corresponding overlapping classes were removed from the reference dataset. For example, when novelty detection was performed based on the Caltech 256, classes appearing in both Caltech 256 and ImageNet were removed from the ImageNet dataset prior to training.

The Area Under the Curve (AUC) and the Equal Error Rate (EER) of the Receiver Operating Characteristic (ROC) Curve are used to measure the performance of different methods. The reported performance figures in this paper are the average AUC figures obtained by considering multiple classes available in the dataset. In all of our experiments, Euclidean distance was used to evaluate the similarity between a test image and the stored templates. In all experiments, the performance of the proposed method was evaluated based on both the Alexnet [21] and the VGG16 [34] architectures. In all experimental tasks, the following experiments were conducted.

Alexnet Features and VGG16 Features (Baseline). Proposed testing mechanism is applied on the $fc7$ Alexnet features and the $fc7$ VGG16 features, respectively.

Alexnet Binary and VGG16 Binary (Baseline). A binary CNN is trained by having ImageNet samples and one-class image samples as the two classes using Alexnet and VGG16 architectures, respectively. Proposed testing mechanism is applied on the $fc7$ features.

One-class SVM [33]. One-class SVM is trained on the image pixels of the training data. The SVM classifier output is used to perform verification.

Autoencoder [14]. Network architecture proposed in [14] is used to learn a representation of the data. Reconstruction loss is used to perform verification.

Ours (Alexnet) and ours (VGG16). Proposed method applied with Alexnet and VGG16 network backbone architectures. The $fc7$ features are used during testing.

In addition to these baselines, in each experiment we report the performance of other task specific methods.
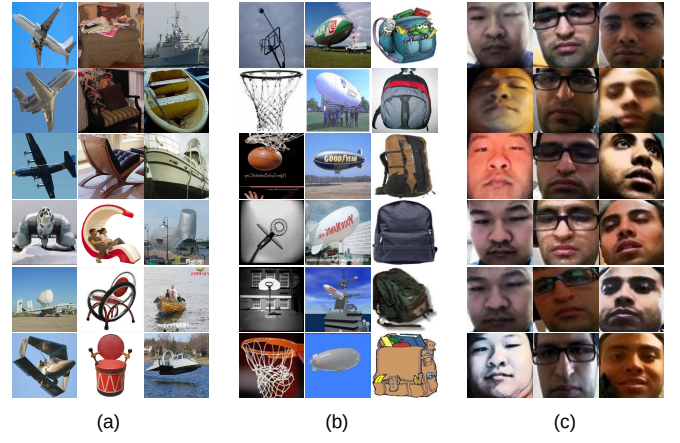

Fig. 7. Sample images from datasets used for evaluation. (a) Abnormal image detection : Top three rows are normal images taken from PASCAL dataset. Bottom three rows are abnormal images taken from Abnormal 1001 dataset. (b) Novelty detection: images from Caltech 256 dataset. (c) Active Authentication: images taken from UMDAA02 dataset. In (b) and (c) columns are formed by images from a single class.

4.2 Results

Abnormal Image Detection: The goal in abnormal image detection is to detect abnormal images when the classifier is trained using a set of normal images of the corresponding class. Since the nature of abnormality is unknown a priori, training is carried out using a single class (images belonging to the normal class). The 1001 Abnormal Objects Dataset [30] contains 1001 abnormal images belonging to six classes which are originally found in the PASCAL [11] dataset. Six classes considered in the dataset are Airplane, Boat, Car, Chair, Motorbike and Sofa. Each class has at least one hundred abnormal images in the dataset. A sample set of abnormal images and the corresponding normal images in the PASCAL dataset are show in Figure 7(a). Abnormality of images has been judged based on human responses received on the Amazon Mechanical Turk. We compare the performance of abnormal detection of the proposed framework with conventional CNN schemes and with the comparisons presented in [30]. It should be noted that our testing procedure is consistent with the protocol used in [30].

Results corresponding to this experiment are shown in Table 2. Adjusted graphical model presented in [30] has outperformed methods based on traditional deep features. The introduction of the proposed framework has improved the performance in Alexnet almost by a margin of 14%. Proposed method based on VGG produces the best performance on this dataset by introducing a 4.5% of an improvement as compared with the Adjusted Graphical Method proposed in [30].

One-Class Novelty Detection: In one-class novelty detection, the goal is to assess the novelty of a new sample based on previously observed samples. Since novel examples do not exist prior to test time, training is carried out using one-class learning principles. In the previous works [4], [10], the performance of novelty detection has been assessed based on different classes of the ImageNet and the Caltech 256 datasets. Since all CNNs used in our work have been trained using the ImageNet dataset, we use the Caltech 256 dataset to evaluate the performance of one-class novelty detection. The Caltech 256 dataset contains images belonging to 256 classes with total of 30607 images. In our experiments, each single class

TABLE 2
Abnormal image detection results on the 1001 Abnormal Objects
dataset.

Method	AUC (Std. Dev.)	EER (Std. Dev.)
One Class SVM [33]	0.598	-
Two Class SVM	0.866	-
Graphical Model [30]	0.870	-
Adjusted Graphical Model [30]	0.911	-
Autoencoder [14]	0.674 (0.120)	0.379 (0.097)
Alexnet Features	0.790 (0.074)	0.294 (0.075)
VGG16 Features	0.847 (0.074)	0.230 (0.074)
Alexnet Binary	0.621 (0.153)	0.401 (0.048)
VGG16 Binary	0.848 (0.081)	0.262 (0.027)
DOC Alexnet (ours)	0.930 (0.032)	0.146 (0.039)
DOC VGG16 (ours)	**0.956 (0.031)**	**0.104 (0.050)**

was considered separately and all other classes were considered as
alien. Sample images belonging to three classes in the dataset
are shown in Figure 7 (b). First, consistent with the protocol
described in [10], AUC of 20 random repetitions were evaluated
by considering the *American Flag* class as the known class and
by considering boom-box, bulldozer and can-non classes as alien
classes. Results corresponding to different methods are tabulated
in Table 3.

In order to evaluate the robustness of our method, we carried
out an additional test involving all classes of the Caltech 256
dataset. In this test, first a single class is chosen to be the
enrolled class. Then, the effectiveness of the learned classifier was
evaluated by considering samples from all other 255 classes. We
did 40 iterations of the same experiment by considering first 40
classes of the Caltech 256 dataset one at a time as the enrolled
class. Since there are 255 alien classes in this test as opposed to the
first test, where there were only three alien classes, performance is
expected to be lower than in the former. Results of this experiment
are tabulated in Table 4.

TABLE 3
Novelty detection results on the Caltech 256 where *American Flag*
class is taken as the known class.

Method	AUC (Std. Dev.)	EER (Std. Dev.)
One Class SVM [33]	0.606 (0.003)	-
KNFST [4]	0.575 (0.004)	-
Oc-KNFD [10]	0.619 (0.003)	-
Autoencoder [14]	0.532(0.003)	0.481(0.003)
Alexnet Features	0.811 (0.003)	0.280 (0.028)
VGG16 Features	0.951 (0.023)	0.124 (0.035)
Alexnet Binary	0.920 (0.026)	0.156 (0.031)
VGG16 Binary	0.997 (0.001)	0.023 (0.003)
DOC Alexnet (ours)	0.930 (0.005)	0.148 (0.008)
DOC VGG16 (ours)	**0.999 (0.001)**	**0.018 (0.004)**

It is evident from the results in Table 3 that a significant
improvement is obtained in the proposed method compared to
previously proposed methods. However, as shown in Table 4 this
performance is not limited just to a American Flag. Approximately
the same level of performance is seen across all classes in the
Caltech 256 dataset. Proposed method has improved the perfor-
mance of Alexnet by nearly 13% where as the improvement the
proposed method has had on VGG16 is around 9%. It is interesting
to note that binary CNN classifier based on the VGG framework
has recorded performance very close to the proposed method in
this task (difference in performance is about 1%). This is due to
the fact that both ImageNet and Caltech 256 databases contain
similar object classes. Therefore, in this particular case, ImageNet

samples are a good representative of novel object classes present
in Caltech 256. As a result of this special situation, binary CNN is
able to produce results on par with the proposed method. However,
this result does not hold true in general as evident from other two
experiments.

TABLE 4
Average Novelty detection results on the Caltech 256 dataset.

Method	AUC (Std. Dev.)	EER (Std. Dev.)
One Class SVM [33]	0.531 (0.120)	0.452 (0.098)
Autoencoder [14]	0.623 (0.072)	0.415 (0.056)
Alexnet Features	0.820 (0.062)	0.260 (0.059)
VGG16 Features	0.897 (0.050)	0.185 (0.057)
Alexnet Binary	0.860 (0.065)	0.218 (0.062)
VGG16 Binary	0.972 (0.024)	0.081 (0.042)
DOC Alexnet (ours)	0.959 (0.021)	0.103 (0.033)
DOC VGG16 (ours)	**0.981 (0.022)**	**0.061 (0.039)**

Active Authentication (AA): In the final set of tests, we evaluate
the performance of different methods on the UMDAA-02 mobile
AA dataset [22]. The UMDAA-02 dataset contains multi-modal
sensor observations captured over a span of two months from
48 users for the problem of continuous authentication. In this
experiment, we only use the face images of users collected by
the front-facing camera of the mobile device. The UMDAA-
02 dataset is a highly challenging dataset with large amount of
intra-class variation including pose, illumination and appearance
variations. Sample images from the UMDAA-02 dataset are shown
in Figure 7 (c). As a result of these high degrees of variations, in
some cases the inter-class distance between different classes seem
to be comparatively lower making recognition challenging.

During testing, we considered first 13 users taking one user
at a time to represent the enrolled class where all the other users
were considered to be alien. The performance of different methods
on this dataset is tabulated in Table 5.

TABLE 5
Active Authentication results on the UMDAA-02 dataset.

Method	AUC (Std. Dev.)	EER (Std. Dev.)
One Class SVM [33]	0.594 (0.070)	0.453 (0.071)
Autoencoder [14]	0.643 (0.074)	0.407 (0.057)
Alexnet Features	0.708 (0.060)	0.354 (0.049)
VGG16 Features	0.748 (0.082)	0.316 (0.067)
Alexnet Binary	0.627 (0.128)	0.405 (0.103)
VGG16 Binary	0.687 (0.086)	0.362 (0.068)
DOC Alexnet (ours)	0.786 (0.061)	0.287 (0.054)
DOC VGG16 (ours)	**0.810 (0.067)**	**0.256 (0.067)**

Recognition results are comparatively lower for this task
compared to the other tasks considered in this paper. This is both
due to the nature of the application and the dataset. However,
similar to the other cases, there is a significant performance
improvement in proposed method compared to the conventional
CNN-based methods. In the case of Alexnet, improvement induced
by the proposed method is nearly 8% whereas it is around 6% for
VGG16. The best performance is obtained by the proposed method
based on the VGG16 network.

4.3 Discussion

Analysis on mis-classifications: The proposed method produces
better separation between the class under consideration and alien

samples as presented in the results section. However, it is interesting to investigate on what conditions the proposed method fails. Shown in Figure 8 are a few cases where the proposed method produced erroneous detections for the problem of one-class novelty detection with respect to the *American Flag* class (in this experiment, all other classes in Caltech256 dataset were used as alien classes). Here, detection threshold has been selected as $\delta = 0$. Mean detection scores for *American Flag* and alien images were 0.0398 and 8.8884, respectively.

As can be see from Figure 8, in majority of false negative cases, the American Flag either appears in the background of the image or it is too closer to clearly identify its characteristics. On the other hand, false positive images either predominantly have American flag colors or the texture of a waving flag. It should be noted that the nature of mis-classifications obtained in this experiment are very similar to that of multi-class CNN-based classification.

| 1.057 | 0.8088 | | 0.000 | 0.000 | 0.000 |
| 0.4241 | 0.3080 | | 0.000 | 0.000 | 0.000 |

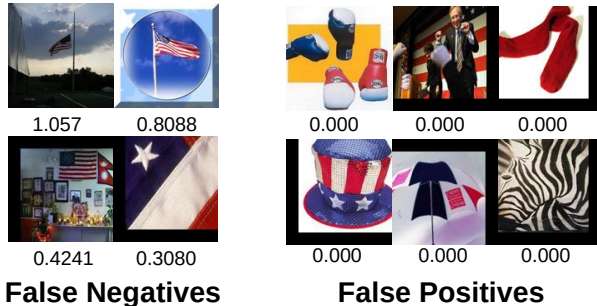**False Negatives**   **False Positives**

Fig. 8. Sample false detections for the one-class problem of novelty detection where the class of interest is *American Flag* (Detection threshold $\delta = 0$). Obtained Euclidean distance scores are also shown in the figure.

Using a subset of the reference dataset: In practice, the reference dataset is often enormous in size. For example, the ImageNet dataset has in excess of one million images. Therefore, using the whole reference dataset for transfer learning may be inconvenient. Due to the low number of training iterations required, it is possible to use a subset of the original reference dataset in place of the reference dataset without causing over-fitting. In our experiments, training of the reference network was done using the validation set of the ImageNet dataset. Recall that initially, both networks are loaded with pre-trained models. It should be noted that these pre-trained models have to be trained using the whole reference dataset. Otherwise, the resulting network will have poor generalization properties.

Number of training iterations: In an event when only a subset of the original reference dataset is used, the training process should be closely monitored. It is best if training can be terminated as soon as the composite loss converges. Training the network long after composite loss has converged could result in inferior features due to over-fitting. This is the trade-off of using a subset of the reference dataset. In our experiments, convergence occurred around 2 epochs for all test cases (Figure 6). We used a fixed number of iterations (700) for each dataset in our experiments.

Effect of number of templates: In all conducted experiments, we fixed the number of templates used for recognition to 40. In order to analyze the effect of template size on the performance of

our method, we conducted an experiment by varying the template size. We considered two cases: first, the novelty detection problem related to the *American Flag* (all other classes in Caltech256 dataset were used as alien classes), where the recognition rates were very high at 98%; secondly, the AA problem where the recognition results were modest. We considered Ph01USER002 from the UMDAA-02 dataset for the study on AA. We carried out twenty iterations of testing for each case. The obtained results are tabulated in Table 6.

TABLE 6
Mean AUC (with standard deviation values in brackets) obtained for different template sizes.

Number of templates	1	5	10	20	30	40
American Flag (Caltech 256)	0.987 (0.0034)	**0.988** (0.0032)	0.987 (0.0030)	0.988 (0.0038)	0.988 (0.0029)	0.988 (0.0045)
Ph01USER002 (UMDAA02)	0.762 (0.0226)	0.788 (0.0270)	0.806 (0.0134)	0.787 (0.0262)	0.821 (0.0165)	**0.823** (0.0168)

According to the results in Table 6, it appears that when the proposed method is able to isolate a class sufficiently, as in the case of novelty detection, the choice of the number of templates is not important. Note that even a single template can generate significantly accurate results. However, this is not the case for AA. Reported relatively lower AUC values in testing suggests that all faces of different users lie in a smaller subspace. In such a scenario, using more templates have generated better AUC values.

5 CONCLUSION

We introduced a deep learning solution for the problem of one-class classification, where only training samples of a single class are available during training. We proposed a feature learning scheme that engineers class-specific features that are generically discriminative. To facilitate the learning process, we proposed two loss functions *descriptiveness loss* and *compactness loss* with a CNN network structure. Proposed network structure could be based on any CNN backbone of choice. The effectiveness of the proposed method is shown in results for Alexnet and VGG16-based backbone architectures. The performance of the proposed method is tested on publicly available datasets for abnormal image detection, novelty detection and face-based mobile active authentication. The proposed method obtained the state-of-the-art performance in each test case.

ACKNOWLEDGEMENT

APPENDIX A: DERIVATIONS

Batch-variance Loss is a Scaled Version of Sample Variance: Consider the definition of batch-variance loss defined as,

$l_b = \frac{1}{nk} \sum_{i=1}^{n} \mathbf{z_i}^T \mathbf{z_i}$ where, $\mathbf{z_i} = \left[\mathbf{x_i} - \frac{1}{n-1} \sum_{j \neq i} \mathbf{x_j} \right]$. Rearranging terms in $\mathbf{z_i}$,

$$\mathbf{z_i} = \left[\mathbf{x_i} - \frac{1}{n-1} \sum_{j=1}^{n} \mathbf{x_j} + \frac{1}{n-1} \mathbf{x_i} \right]$$

$$\mathbf{z_i} = \left[\frac{n}{n-1} \mathbf{x_i} - \frac{1}{n-1} \sum_{j=1}^{n} \mathbf{x_j} \right]$$

$$\mathbf{z_i} = \frac{n}{n-1}\left[\mathbf{x_i} - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x_j}\right]$$

$$\mathbf{z_i}^T\mathbf{z_i} = \frac{n^2}{(n-1)^2}\left[\mathbf{x_i} - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x_j}\right]^T\left[\mathbf{x_i} - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x_j}\right]$$

But, $\left[\mathbf{x_i} - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x_j}\right]^T\left[\mathbf{x_i} - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x_j}\right]$ is the sample variance σ_i^2. Therefore,

$$l_b = \frac{1}{nk}\sum_{i=1}^{n}\frac{n^2\sigma_i^2}{(n-1)^2}$$

Therefore, $l_b = \beta\sigma^2$, where $\beta = \frac{n^2}{k(n-1)^2}$ is a constant and σ^2 is the average sample variance.

Backpropagation of Batch-variance Loss:

Consider the definition of batch variance loss l_b,
$l_b = \frac{1}{nk}\sum_{i=1}^{n}\mathbf{z_i}^T\mathbf{z_i}$ where, $\mathbf{z_i} = \mathbf{x_i} - \mathbf{m_i}$.
From the definition of the inner product,
$\mathbf{z_i}^T\mathbf{z_i} = \sum_{j=1}^{k}z_{ij}^2$. Therefore, l_b can be written as,

$$l_b = \frac{1}{nk}\sum_{i=1}^{n}\sum_{l=1}^{k}(x_{il} - m_{il})^2.$$

Taking partial derivatives of l_b with respect to x_{ij}. By chain rule we obtain,

$$\frac{\partial l_b}{\partial x_{ij}} = \frac{2}{nk}\sum_{l=1}^{k}x_{il} - m_{il} \times \frac{\partial x_{il} - m_{il}}{\partial x_{ij}}.$$

Note that $\frac{\partial x_{ij} - m_{ij}}{\partial x_{ij}} = 1$ when $j = l$. Otherwise, $\frac{\partial x_{ij} - m_{ij}}{\partial x_{ij}} = -\frac{\partial m_{ij}}{\partial x_{ij}} = \frac{-1}{n-1}$.

$$\frac{\partial l_b}{\partial x_{ij}} = \frac{2}{nk}\left[x_{ij} - m_{ij} - \frac{1}{n-1}\sum_{l\neq j}x_{il} - m_{il}\right].$$

$$\frac{\partial l_b}{\partial x_{ij}} = \frac{2}{nk}\left[\frac{n}{n-1}x_{ij} - m_{ij} - \frac{1}{n-1}\sum_{l=1}^{n}x_{il} - m_{il}\right].$$

$$\frac{\partial l_b}{\partial x_{ij}} = \frac{2}{(n-1)nk}\left[n \times (x_{ij} - m_{ij}) - \sum_{l=1}^{n}(x_{il} - m_{il})\right].$$

APPENDIX B : DETAILED NETWORK ARCHITECTURES

Shown in Figure 9 are the adaptations of the proposed method to the existing CNN architectures- (a) Alexnet and (b) VGG16, respectively. We have used these two architectures to carry out all experiments in this paper. In both architectures, two images from the target dataset t and the reference dataset r are fed into the network to evaluate cross entropy loss and batch-variance loss. Weights of convolution and fully-connected layers are shared between the two branches of the network. For all experiments, stochastic gradient descent algorithm is used with a learning rate of 5×10^{-5} and a weight decay of 0.0005.
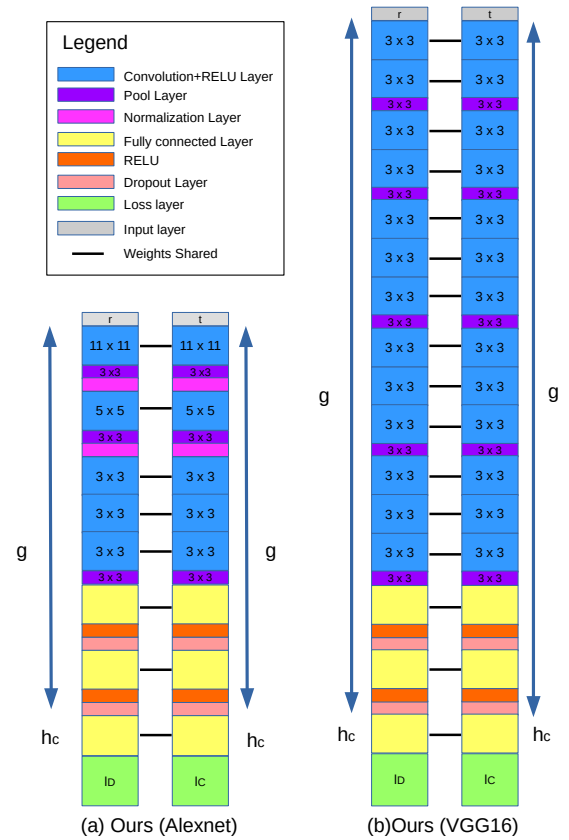

Fig. 9. CNN architectures based on Alexnet and VGG16 backbones for the proposed method.

REFERENCES

[1] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[2] A. Bendale and T. E. Boult. Towards open set deep networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1563–1572, 2016.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.

[4] P. Bodesheim, A. Freytag, E. Rodner, M. Kemmler, and J. Denzler. Kernel null space methods for novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[6] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 539–546, June 2005.

[7] D. A. Clifton, S. Hugueny, and L. Tarassenko. Novelty detection with multivariate extreme value statistics. *Journal of signal processing systems*, 65(3):371–389, 2011.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 647–655, Bejing, China, 22–24 Jun 2014.

[10] F. Dufrenois and J. Noyer. A null space based one class kernel fisher discriminant. In *International Joint Conference on Neural Networks (IJCNN)*, 2016.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies

for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[13] J. A. Gubner. *Probability and Random Processes for Electrical and Computer Engineers*. 2006.

[14] R. Hadsell, S. Chopra, and Y. Lecun. Dimensionality reduction by learning an invariant mapping. In *In Proc. Computer Vision and Pattern Recognition Conference (CVPR06*. IEEE Press, 2006.

[15] H. He and Y. Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley-IEEE Press, 1st edition, 2013.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, June 2016.

[17] P. Hu and D. Ramanan. Finding tiny faces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, pages 675–678, 2014.

[19] M. Kemmler, E. Rodner, E.-S. Wacker, and J. Denzler. One-class classification with gaussian processes. *Pattern Recogn.*, 46(12):3507–3518, 2013.

[20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[22] U. Mahbub, S. Sakar, V. Patel, and R. Chellappa. Active authentication for smartphones: A challenge data set and benchmark results. In *IEEE International Conference on Biometrics: Theory, Applications and Systems*, Sept 2016.

[23] M. Markou and S. Singh. Novelty detection: a review – part 1: statistical approaches. *Signal Processing*, 83(12):2481 – 2497, 2003.

[24] M. Markou and S. Singh. Novelty detection: a review – part 2: neural network based approaches. *Signal Processing*, 83(12):2499 – 2521, 2003.

[25] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello. Continuous user authentication on mobile devices: Recent progress and remaining challenges. *IEEE Signal Processing Magazine*, 33(4):49–61, July 2016.

[26] E. Platzer, H. Se, J. Ngele, K.-H. Wehking, and J. Denzler. On the suitability of different features for anomaly detection in wire ropes. In *Computer Vision, Imaging and Computer Graphics: Theory and Applications*, volume 68, pages 296–308, 2010.

[27] S. J. Roberts. Novelty detection using extreme value statistics. *IEE Proceedings-Vision, Image and Signal Processing*, 146(3):124–129, 1999.

[28] E. Rodner, B. Barz, Y. Guanche, M. Flach, M. D. Mahecha, P. Bodesheim, M. Reichstein, and J. Denzler. Maximally divergent intervals for anomaly detection. *CoRR*, abs/1610.06761, 2016.

[29] E. Rodner, E. sabrina Wacker, M. Kemmler, and J. Denzler. One-class classification for anomaly detection in wire ropes with gaussian processes in a few lines of code. In *In: Conference on Machine Vision Applications (MVA)*, pages 219–222, 2011.

[30] B. Saleh, A. Farhadi, and A. Elgammal. Object-centric anomaly detection by attribute-based reasoning. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 787–794, 2013.

[31] P. Samangouei and R. Chellappa. Convolutional neural networks for attribute-based active authentication on mobile devices. In *IEEE International Conference on Biometrics: Theory, Applications and Systems*, Sept 2016.

[32] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult. Towards open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36, July 2013.

[33] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, 2001.

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[35] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1):45–66, 2004.

[36] L. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[37] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103, 2008.

[38] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[39] S. Yang, P. Luo, C. C. Loy, and X. Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[40] H. Zhang and V. M. Patel. Sparse representation-based open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1690–1696, Aug 2017.