Novelty Detection in MultiClass Scenarios with Incomplete Set of Class Labels

Nomi Vinokurov

Submitted in partial fulfillment of the requirements
of the degree of Master of Science

Under the supervision of
Prof. Daphna Weinshall

May 1, 2016

Rachel and Selim Benin
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Israel

Abstract

In this work we address the problem of novelty detection in multiclass scenarios where some class labels are missing from the training set. As opposed to other approaches, we learn novelty representation in a multiclass classifier context, instead of learning only the known data.

Our method is based on a novelty representation that we defined. This representation is based on the initial assignment of confidence values, which measures the affinity between a new test point and each known class. We first compare the values of the two top elements in this vector of confidence values. In the heart of our method lies the training of an ensemble of classifiers, each trained to discriminate known from novel classes, based on some partition of the training data into presumed-known and presumed-novel classes. Our final *novelty score* is derived from the output of this ensemble of classifiers.

We evaluated our method on two datasets of images containing a relatively large number of classes - the Caltech-256 and Cifar-100 datasets. We compared our method to 3 alternative methods which represent commonly used approaches, including the one-class SVM, novelty based on k-NN, novelty based on maximal confidence, and the recent KN-FST method. The results show a very clear and marked advantage for our method over all alternative methods, in an experimental setup where class labels are missing during training.

זיהוי אובייקטים חדשים בתרחיש מרובה מחלקות

נעמי וינוקורוב

הוגש כמילוי חלקי של החובות לתואר מוסמך במדעים

מנחה: פרופ' דפנה וינשל

ניסן תשע"ו

בית הספר להנדסה ומדעי המחשב
על שם רחל וסלים בנין
האוניברסיטה העברית בירושלים
ישראל

תקציר

בעבודה זו אנו מתייחסים לבעית זיהוי אובייקט חדש בתרחיש בו יש הרבה מחלקות מוכרות ומחלקות חדשות, ללא יצוג בשלב האימון. בשונה מהרבה עבודות בתחום שמתמקדות בלמידת התנהגות של האובייקטים המוכרים בלבד, אנו מנסים ללמוד כיצד מתנהג אובייקט חדש, לעומת ההתנהגות של אובייקט השייך למחקלה מוכרת.

התייחסנו לתרחיש בו יש קבוצת אובייקטים שידוע שכולם שייכים לאותה מחלקה, אך לא ידוע האם מחלקה זו היא מוכרת או חדשה. במקרה שבו גודל הקבוצה הוא אחד, זו הבעיה המקורית של זיהוי אובייקט חדש. מתי יתכן תרחיש בו גודל הקבוצה גדולה מאחד? בדגימה מתוך מושבת חיידקים- מקור כל החיידקים מאותו חיידק, כלומר כל החיידקים זהים ולכן שייכים לאותה מחלקה, אך יתכן כי זו מחלקה חדשה שהמסווג לא מכיר. דוגמא נוספת היא כאשר עובדים עם סרט ממצלמת אבטחה, פריימים צמודים בזמן הם למעשה כמה תמונות של אותו האובייקט.

השיטה שלנו מבוססת על ייצוג מסויים, אותו הגדרנו ואותו אנחנו לומדים עבור אובייקטים חדשים ואובייקטים השייכים למחלקות המוכרות. ראשית אנו מאמנים מסווג רב מחלקתי, המסווג אובייקטים בין המחלקות המוכרות. בהינתן אובייקט מסויים אנחנו מסתכלים על ערכי הניבוי המתקבלים ע״י המסווג המאומן. אנחנו מחשבים את היחס שבין שני ערכי הניבוי הגבוהים ביותר. במקביל אנחנו מסתכלים על אותו היחס המתקבל עבור אובייקטים משלב האימון שידוע שהם השייכים למחלקה אותה היינו מנבאים עבור האובייקט המדובר.

כדי לאפשר למידה של יצוג האובייקטים החדשים ולהמנע מתיוג אובייקטים אלו כחדשים בשלב המבחן (מאחר שאובייקטים שנחשפנו אליהם בשלב האימון כבר אינם חדשים), אנו מאמנים אנסמבל של מסווגים בינאריים לסווג בין אובייקטים חדשים לאובייקטים מוכרים על בסיס הייצוג אותו הגדרנו, כל אחד עם חלוקה שונה של כל המחלקות המוכרות למחלקות ׳מוכרות- זמניות׳ ומחלקות ׳חדשות זמניות׳ לצורך האימון. לבסוף המדד שבו אנו משתמשים הוא מספר המסווגים שתייגו דוגמא ספציפית כחדשה (במקום להסתמך על ערך הכרעה של מסווג בינארי יחיד).

אימתנו את השיטה שלנו על שני דאטא-סטים: קלטק 256 וסיפר 100. השוונו את הביצועים שלנו לביצועים של 3 שיטות נוספות, המייצגות שיטות סטנדרטיות לזיהוי אובייקטים חדשים. שיטת השכנים הקרובים (k-nn), one class SVM, ומיקסום הנראות. בנוסף השוונו את עצמו לשיטה הנקראת KNFST, המתמקדת בתרחיש דומה לשלנו- כאשר הרבה מחלקות הן מחלקות מוכרות, ויש מחלקות שאינן מוכרות. התוצאות מראות שהשיטה שלנו עובדת טוב יותר.

Acknowledgments

I would like to thank my supervisor, Prof. Daphna Weinshal, who had guided me throughout the research and helped me to discover the exciting field of novelty detection. I am thankful and indebted to her for introducing me to her unique way of thinking, and for encouraging me to explore new questions and directions, and finally to come up with innovative ideas.

I would also like to thank my friends and peers- Tamar Elazari, Lior Bar, Talia Tron, Hezi Reshef, Nitzan Guberman, Galit Bari, and Daniel Hadar for discussing with me the intermediary results, offering a different perspective, and facilitating the learning process.

Finally, I am greatly thankful to my family for their support. Most especially, I thank my husband Amitai and my son Dagan for the inspiration that they gave me along the way.

Contents

Chapter 1

Introduction

Novelty Detection is an issue to reckon with when using almost any machine learning tool. Machine learning tools typically attempt to generate predictions for future events based on past events in some training dataset. The role of novelty detection is to warn the system when events in the test set are inherently different from those observed in the training set, and therefore the system should not attempt to label them with predictors based on past events. In this guise novelty detection is often used as synonym to outlier or anomaly detection. Common applications include such tasks as the identification of system malfunctioning, or the identification of unexpected behavior in the context of a security system or health monitoring.

With the emerging world of big-data where multiclass classification scenarios with many class labels are becoming more common, another twist on the kind of events which are novel to the system by definition presents itself quite frequently - when whole classes are missing from the training set [9]. These observations are not anomalous nor are they outliers, but rather they belong to *novel* classes for which the system has not been given any training examples. In real-life scenarios this may happen more often than not, due possibly to peculiarities of the labeling protocol, or due to appearance of a completely new classes in the world after the training stage. In movement ecology, for example, observers can only label observations when the birds are visible to people on the ground and during day time; measurements which occur away from the observers cannot be labeled, although measurements are being continuously collected. Another example is in a face detection scenario. During the training stage the classifier learns a finite set of face classes, while in the test stage there might be an appearance of a face of a new person.

In this work we propose a method for the detection of novel classes. We observe that unlike the problem of novelty detection as used for the detection of outliers or anomalies, in the present scenario one can learn from the training data something about what best discriminates known from unknown classes, and attempt to use this learned insight to identify datapoints emerging from novel classes.

In a typical scenario where novelty detection is to be employed, the training data is composed mostly of points sampled from a single well-sampled class - the *normal* class, with possibly a few examples marked as *abnormal*. The *abnormal* points do not provide a sufficiently representative sample of the under-sampled *abnormal* class. Novelty detection methods may attempt to learn a model of normality, a model against which new test examples will be compared to obtain some form of novelty score. This score is compared, in turn, against a threshold to determine novelty. Consequently method efficacy is usually evaluated with the Receiver Operating Characteristic (ROC) curve of the novelty score, using quantities such as the Area Under the Curve (AUC) or Equal Error Rate (EER) of the ROC curve.

Many methods have been developed in order to provide a *novelty* score. Some methods are probabilistic involving the density estimation of the *normal* class; the *novelty* score measures the likelihood that a test point comes from the same distribution as the training set (e.g., [12, 7]). Other methods rely on a notion of distance between points, and measure novelty by the distance (or similarity) between a test point and the training set (e.g., [13]). Yet another family of methods construct a model of the *normal* class, generating for each test point the closest estimate which the model can produce; the reconstruction error, which is the difference between the output of the reconstruction method and the actual test point, provides the *novelty* score (e.g., [1]). Finally, some methods follow the discriminative approach dominant in machine learning, and construct a boundary around the *normal* class which is used to separate between *normal* and *novel* points (e.g., [21]).

Recently, [4, 3] described a novelty detection method based on kernel null space (denoted KNFST). This method uses the kernel trick to project the training points into the null space of known classes. Subsequently the *novelty score* is the distance in the projected space between a test point and the known classes which are represented by singletons. Like ours, this method was designed to detect novel classes rather than outliers or anomalies, and it was evaluated using similar databases of images as we use here. We note that with big data and specifically with many known classes (hundreds of them), projection to the null-space of all classes may eventually deplete the remaining degrees of freedom (figuratively speaking) too much. This decrease in performance can be pronounced even for 60 known classes, as the results in [3] show.

A thorough recent review of the state of the art in novelty detection can be found in [18]. In the experiments described below in chapter 4, we compare our method to a few simple and representative methods from this vast literature - the discriminative one class SVM [21], and novelty based on k-nearest-neighbor [8]. We also compare our method to KNFST described above.

In the multiclass scenarios, the most relevant methods are usually tied to the notion of *reject*. In the context of statistical pattern recognition, a pattern is to be rejected when its highest posterior probability to be assigned to any of the known classes is less than a threshold [7]. This notion has been further generalized to include different kinds of reject [9], including *distance reject* - a notion quite similar to novelty, and *ambiguity reject* - a different notion identifying ambiguous patterns which can be assigned to more than one class. In [16], a number of decision procedures are discussed, following an initial assignment procedure where each test point is assigned a vector of soft labels for each class.

We compare our method to *distance reject* based on comparing the maximal confidence to a threshold; this simple baseline method performs rather similarly to more complicated decision procedures when limited to *distance reject* [16].

Our method, described in chapter 3, starts by assuming the availability of a vector of soft label assignments for each point at test time. This representation can be obtained from almost any multiclass classification algorithm, including Convolution Neural Network [15] or multiclass SVM [5]. First, for each test point we compute a *raw novelty score*, which is based on ordering the vector of soft assignments and comparing the values of the best and second best assignments (by difference or ratio). This raw *novelty score* is different from what is commonly used for novelty detection (cf. [16]), which is typically the value of the best assignment - the most likely one when the soft labels correspond to actual probabilities [7]. This score bears some similarity to uncertainty criteria used for active learning, as in [20]. For novelty detection, when relying on a discriminative multiclass classifier such as one-vs-all multiclass SVM, we note that this score performs well (see Fig. 3.3), possibly because it involves the comparison between the confidence of the two top assignments which can identify points far away from the decision boundaries.

To go beyond the *raw novelty score*, we recall that it is new class labels which we seek to identify, and therefore we can use the training data to learn a classifier for this purpose. Specifically, we construct an ensemble of L classifiers $h_l(x)$, each trained to discriminate between novel and known classes based on some random artificial partition of the training set into novel and known classes. Each classifier $h_l(x)$ is trained to discriminate between the *raw novelty score* of x, and the average raw novelty score of the points in the class which is chosen as top assignment for x. The final *novelty score* is the count in $[0 \dots L]$ of classifiers in the ensemble which identified x as novel.

Our proposed framework goes beyond the usual novelty detection framework, adopting a scenario which is relevant to our days of big data, where novelty detection is not only meant to detect faulty systems but also detect valid but previously unseen events. Thus we address a more elaborate scenario, where one knows that a group of test points belongs to the same (known or novel) class. The size of the group is a parameter s; for $s = 1$, this scenario reduces to the usual novelty detection framework. Why is this relevant to real life? Imagine collecting a sample of s points from a bacteria colony; it is known that all s points originated from the same bacteria, and the required decision is whether this sample belongs to a known bacteria or a new strand.

The rest of this work is organized as follows. In chapter 2 we review previous novelty detection approaches. In chapter 3 we describe our proposed method, with its two separate layers, and provide some analysis. In chapter 4 we describe the experimental evaluation of our method, comparing it to representative novelty detection methods from the literature reviewed above. Unlike the experimental evaluation described in [8], we focus our evaluation on large databases with 100 classes or more. Our method is shown to very significantly outperform other methods, including a method which uses only the aforementioned *raw novelty score* for novelty detection.

Chapter 2

Related works

In this chapter, we briefly introduce some novelty detection approaches. As stated in previous part, novelty detection is often used for outlier or anomaly detection. Thus, in the following we will overview mostly methods that have been used for these purposes.

2.1 Probabilistic approach

Probabilistic approach for novelty detection is based on estimating the generative probability density function of the known data. The resultant distribution is threshold to define the boundaries of normality in the data space and verify whether a test sample comes from the same distribution. Training data are assumed to be generated from some underlying probability distribution. These data points are used to estimate the distribution that represents a model of normality. Given a test point these methods measures the likelihood that a test point comes from estimated distribution. A very common form of distribution for continuous variables is Gaussian, the parameters of which are estimated from the given training set using maximum likelihood. More complex forms of data distribution frequently used are Gaussian mixture models (GMMs). The parameters of this model may be estimated using maximum likelihood methods via optimization algorithms such as expectation maximization [2]. Other mixtures of different types of distribution, such as gamma, and Poisson are used as well [18]. However in the absence of prior information regarding the form of the underlying distribution of the data, the Gaussian distribution is often used because of its convenient analytical properties when determine the location of a novelty detection. The Grubbs' test, for example, assumes a Gaussian distribution for training data and computes the distance of the test data points from the estimated mean defining any point with distance above a certain threshold to be an outlier.[12]

2.1.1 Optimum rejection rule

Chow was interested in the error-reject tradeoff and the optimum decision rule, minimizing the error probability for a given reject probability [7, 6]. An error is a misrecognition, it occurs when a pattern from one class is identified as that of a different class. A reject occurs when the recognition system withholds its decision, and the pattern is rejected. Chow derived an optimum rejection scheme as following. Reject a data point if the maximum of the a posteriori probabilities is less than a user defined threshold. If maximum of the a posteriori probabilities is greater than the threshold, accept and classify the data point according to the class with the maximum a posteriori probability value.

Both error probability and reject probability of the optimum recognition systems are monotonic function of the rejection threshold. Therefore there is a tradeoff between error rate and reject rate. Increasing the threshold results in a higher error probability and lower rejection probability.

2.1.2 Distance and ambiguity reject

Dubuisson and Masson extend Chow's rejection approach and refer to multiclass scenario [9]. They define two kind of rejection, *ambiguity reject* and *distance reject*. *Ambiguity reject* for objects that might fit several classes (this rejection is same as Chows optimum rejection rule). *Ambiguity reject* is not sufficient in order to solve the problem when the number of possible classes is not known. *Distance reject* is defined to solve this problem, for objects that are distributed around known classes or even arise form an unknown class. A point x will be distance rejected if

$$F(x) < t$$

where t is the distance reject threshold.

[16] reviews a family of selection measures and block similarity measures of soft-labels and derives a general class-selective rule which allows in one single step either to distance reject or to classify (one selected class) or to ambiguity reject a pattern, given user specified threshold. [16] intends to avoid total ambiguity rejection whenever a number between 2 and c-1 classes have to be selected, (where c is the number of all known classes), thus a class selective procedure is defined as the seek for the best top-n classes according to

$$\eta^*(X,t) = min_{k \in C} k : \Phi(u(x), k) \leqslant \mathbf{t}$$

where $(u(x))$ is the sorted soft label vector, and $\Phi(u(x))$ is a specific selection measure applied on the sorted soft label vector and $\mathbf{t}$ is a user defined threshold, and C is the set of known classes. There are different selection measures Φ suitable for class-selective rejection instead of total ambiguity rejection. A simple selection measure for example which is defined in [10] is-

$$\Phi(u(x)), k) = u_{(k+1)}(x)/u_{(k)}(x)$$

2.2 Distance- based approach

Distance-based methods do not require a priori knowledge of the data distribution, thus, share some common assumptions with probabilistic approach. Both approaches attempt to characterize the area of the data space occupied by the normal data. Distance-based methods rely on well defined distance metrics to compute the distance between two data points.

2.2.1 K-Nearest neighbour- based methods

K-Nearest-Neighbor (k-NN) approach is based on the assumption that normal data points have close neighbors in the training set, while abnormal points are located far from training points[13]. A point x_i is accepted as normal if the ratio of the distance to its nearest neighbor x_j in the training set and the distance from x_j to the nearest neighbor of x_j is relatively small (a user defined threshold may be used here), otherwise x_i considered as abnormal point. Euclidean distance is a popular choice for continuous attributes. Most of nearest neighbors based techniques identify novel data points globally and are not flexible enough to detect local novelty in data sets that have diverse densities and arbitrary shapes.

In [8], 4 novelty detection methods performance on 10 datasets is compared. The k-NN method appears to outperform other three methods.

The variant that was used in [8] compares the distance between data point x to its k nearest neighbors in the training set $NN_k(x)$, with the distance between $NN_k(x)$ and their own k nearest neighbors in the training set. The *novelty score* of this method was defined as

$$f_{NN_k(x)} = \frac{||x - NN_k(x)||}{||NN_k(x) - NN_k(NN_k(x))||}$$

where x is a data point from the test set, and the operator $NN_k()$ denotes the k nearest neighbors. Low $f_{NN_k(x)}$ value indicates a known point.

K-NN method method is very simple and effective, the drawback is that it needs to store all training data points which are further used to compute the distance between a new unseen point and all training data points.

2.2.2 Clustering based methods

Clustering-based methods for novelty detection, refer to grouping of the normal training data into some clusters. If a new unseen data point belongs to any of these clusters then it belongs to a known class. In this general type of methods the ”known” class is characterized by a small number of prototype points in the data space. The minimum distance from a test point to the nearest prototype is often used to quantify novelty. These methods use different approaches to obtain the prototype locations. A common method to perform clustering is the k-means clustering. Among the clustering based algorithms there are many modifications of the k-means algorithm. One extension, for example, uses wavelets to transform the data and then find dense clusters and outliers in the transformed space [24].

2.3 Margin based technique

One commonly used novelty detection technique is based on Support Vector Machine (SVM), building a decision hyperplane/spherical boundary that separates/encloses the majority of normal data points. A small fraction of the renaming normal data points may be left outside the boundary. Class membership of unknown data is determined by their location with respect to the boundary. These methods are typically insensitive to the specific sampling and density of the normal class, because they describe the normal class boundary, or the domain and not the class density. As with two-class SVMs, novelty detection SVMs determine the location of the novelty boundary using only those data that lie closest to it (in the transformed space) i.e support vectors. All other data form training set are not used when setting the novelty boundary. Hence, the distribution of data in the training set is not considered. Recent works on SVM based novelty detection have been developed along two strands: one-class SVM (OCSVM)[21] and support vector data description (SVDD).[23].

2.3.1 One-classs SVM

OCSVM algorithm proposed in [21] uses a kernel trick to construct a hyperplane that separates the normal data from the origin with maximum margin in a feature space. It supposed to capture regions in input space where the probability lives. OCSVM assumes that a number of training data points fall into some regions and then enforces the regions to be small in a feature space associated to the kernel. The kernel trick makes it much easier to separate the normal data from the origin in a higher dimensional feature space. This algorithm requires fixing a priori a parameter that characterizes the fraction of support vectors and outliers. This enabels OCSVM to be more tolerant to outliers in the ”normal” training data. However, setting this parameter strongly influence the performance of this approach.

2.3.2 Support vector data description

SVDD algorithm proposed by Tax and Duin [23], defines the novelty boundary as being the hypersphere with minimum volume that encloses all or most of the ”normal” training data. Novelty is assessed by determining if a test point lies within the hypersphere. SVDD algorithm also requires fixing a parameter that controls the trade-off between sphere volume and the errors. The main contribution of SVDD is using the kernel trick to make the rigid spheres flexible and is further able to fit irregular shaped normal data distributions. The most popular kernel function is the Gaussian kernel, and the selection of its bandwidth parameters will decide the number of normal points becoming support vectors. Therefore there may be over-fitting of the constructed boundary. The kernel method within SVDD is attractive due to its good results in many novelty detection methods.

Chapter 3

Algorithm

3.1 Notations and *raw novelty score*

Let $X = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ denote a set of N data points (in our experiments these are feature vectors representing images) where $\mathbf{x}_j \in \mathcal{R}^d$. Let $Y = y_1, y_2, \ldots, y_N$ denote the set of corresponding labels in $[k]$, and $\mathcal{S}$ denote a set of test points known to share the same label $y_{\mathcal{S}}$ where $|\mathcal{S}| \geqslant 1$.

Initial representation Each point $\mathbf{x}_j$ is assigned a set of soft labels based on the confidence vector of some multiclass classifier or soft probabilistic assignment. In the following, since we evaluate our methods on two large databases of images - a domain for which the Convolution Neural Network (CNN) architecture has proven most effective for classification [15], we adapt a simple CNN to solve the original multiclass problem defined by the training set. Given X, Y, the output of the CNN classifier is a set of vectors $\mathbf{u}_j \in \mathbb{R}^k, j \in [N]$ containing the vector of activities of the network's k output units. It is also possible to use a multiclass SVM classifier, where $\mathbf{u}_j$ denotes the vector of k margins of the SVM classifier. Either way, $\mathbf{u}_j^i$ denotes the confidence in the assignment of point $\mathbf{x}_j$ to class i.

Raw novelty score Let $o_{\mathcal{S}}$ denote the predicted class assignment of a set of points $\mathcal{S}$ known to be sampled from the same class

$$o_{\mathcal{S}} = \operatorname*{argmax}_{i \in [k]} \operatorname*{mean}_{j \in \mathcal{S}} \mathbf{u}_j^i \tag{3.1}$$

Let $\mathbf{u}(\mathcal{S})$ denote the sorted mean soft labels vector of set $\mathcal{S}$,

$$\mathbf{u}(\mathcal{S}) = sort(\operatorname*{mean}_{j \in \mathcal{S}} \mathbf{u}_j) \tag{3.2}$$

We define the following measure of novelty for set $\mathcal{S}$:

$$\theta_{\mathcal{S}} = \frac{\mathbf{u}(\mathcal{S})_1}{\mathbf{u}(\mathcal{S})_2} \tag{3.3}$$

For a set of points $\mathcal{S}$ (a set which can be a singleton), $\theta_{\mathcal{S}}$ measures the ratio between the confidence in its assignment to the most likely class and the second most likely class. This quotient reflects how ambiguous the choice of $o_{\mathcal{S}}$ is with respect to the second best choice. Presumably, for novel images of known classes $\theta_{\mathcal{S}}$ should be fairly large, while for novel images of novel objects $\theta_{\mathcal{S}}$ will be significantly smaller.

Finally, we define in a similar manner a representative value θ_i for each class label i, using the set $\mathcal{S}_i$ of all training examples labeled i

$$\theta_i = \frac{\mathbf{u}(\mathcal{S}_i)_1}{\mathbf{u}(\mathcal{S}_i)_2} \tag{3.4}$$

The vector $[\theta_i]_{i=1}^{k}$ is estimated from the training data as described in Section 3.3.

3.2 Proposed method to evaluate novelty score

First, we construct L partitions of the set of classes $\mathcal{C}$ in the training data, dividing $\mathcal{C}$ into presumed-known (denoted $\mathcal{C}_K^l$) and presumed-novel (denoted $\mathcal{C}_N^l$) classes, balancing the number of partitions where each class in $\mathcal{C}$ is labeled as novel. For each partition l we train an SVM classifier to discriminate between points in $\mathcal{C}_K^l$ and points in $\mathcal{C}_N^l$. This gives us an ensemble of L classifiers $\mathcal{H} = \{h_l,\ l \in [L]\}$. Note that by construction classifier h_l is trained to erroneously label known classes in $\mathcal{C}_N^l$ as novel, and thus we use $|\mathcal{C}_N^l| << |\mathcal{C}_K^l|$.

Classifier h_l receives two input values, $\theta_{\mathcal{S}}$ and θ_i for $i = o_{\mathcal{S}}$, effectively using $\theta_{o_{\mathcal{S}}}$ to calibrate the value of $\theta_{\mathcal{S}}$. Finally, for set $\mathcal{S}$, its *novelty score* is the count of novel decisions by classifiers in ensemble $\mathcal{H}$, an integer in the range $[0 \ldots L]$. As with other novelty scores, this score is compared to a threshold to determine novelty.

3.3 Details of novelty score evaluation

The following two steps describe the training of a single novelty classifier h_l:

Step 1. partition to presumed-known and presumed-novel, initial representation We start by artificially dividing the training data and temporarily marking approximately 10% of all classes in the training set $\mathcal{C}$ as novel. This set of labels is denoted $\mathcal{C}_N^l$, and the set of remaining labels is denoted $\mathcal{C}_K^l$. Using the training examples with labels in $\mathcal{C}_K^l$, we train a multiclass classifier to solve the corresponding multiclass classification problem with $k' = |\mathcal{C}_K^l|$ classes. Subsequently each point is represented as a vector in $\mathcal{R}^{k'}$, whose $i-th$ element is the confidence of the multiclass classifier in label i. Specifically, since we are working with image databases, we use a simple CNN for multiclass classification, and the activation of the k' output units as the representation vector in $\mathcal{R}^{k'}$. This procedure is described in Algorithm 1.

Algorithm 1 Initial representation of all datapoints for partition l

input:

- X': N' training points with label in $\mathcal{C}_K^l$
- X'': $(N - N')$ training points with label in $\mathcal{C}_N^l$, $X = X' \cup X''$
- Y': class labels of X'

output:

- $Z'^l, Z''^l, \mathcal{F}^l$

```
1: $\mathcal{F}^l$ ←network.train($X'$,$Y'$) {multiclass classifier}
2: for $j = 1$ to $N'$ do
3:     $\mathbf{z}_j'^l \leftarrow \mathcal{F}^l(\mathbf{x}_j')$ {new representation for $X'$}
4: for $j = 1$ to $(N - N')$ do
5:     $\mathbf{z}_j''^l \leftarrow \mathcal{F}^l(\mathbf{x}_j'')$ {new representation for $X''$}
6: return $Z'^l, Z''^l, \mathcal{F}^l$
```

Step 2. Learning the binary classifier for partition l After step 1, let Z'^l denote the set of vectors in $\mathcal{R}^{k'}$ which represent the training data points to be labeled 'known', and Z''^l the set of vectors in $\mathcal{R}^{k'}$ which represent the training data points to be labeled 'novel'. In step 2 we first compute θ_i for each label $i \in \mathcal{C}_K^l$. We then divide Z''^l into disjoint subsets of size s and equal label, and compute for each subset the corresponding pair of values $\{\theta_\mathcal{S}, \theta_{o_\mathcal{S}}\}$. The list of pairs becomes the set of positive examples (with label 'novel') Ψ_P. The set of negative examples (with label 'known') Ψ_N is similarly constructed from Z'^l. Finally we train a linear SVM classifier which, given the pair $\{\theta_\mathcal{S}, \theta_{o_\mathcal{S}}\}$, returns a binary label 'novel' or 'known'. This training procedure is described below in Algorithm 2, and illustrated in Fig. 3.1.

Computing the final novelty score We now use the ensemble of binary classifiers $\mathcal{H} = \{h_l(\theta_\mathcal{S}^l, \theta_{\hat{o}}^l),\ l \in [L]\}$ to compute the novelty score of set $\mathcal{S}$. We first compute $\hat{o} = o_\mathcal{S}$ from (3.1) using a multiclass classifier which has been trained using all the training set and all the labels in $\mathcal{C}$. We extract the set of relevant classifiers from the ensemble $\mathcal{H}$, using every h_l where in partition l, $\hat{o} \in \mathcal{C}_K^l$. Intuitively, this is intended to eliminate classifiers h_l which are trained to identify $\hat{o}$ as novel even though it is a known label, and therefore are likely to harm the decision process regarding class $\mathcal{S}$. The scoring procedure is described in Algorithm 3.

3.4 Algorithm analysis

The benefit of using an ensemble of classifiers Let x denote a point, and y denote its label. Alg. 3 determines the novelty score of a single point x by summing up the binary

Algorithm 2 Training of novelty classifier h_l

```
input:

  • $Z'^l$: $N'$ training points in $\mathcal{R}^{k'}$
  • $Z''^l$: $(N - N')$ training points in $\mathcal{R}^{k'}$
  • $Y'^l$: data labels in $\mathcal{C}_K^l$ for points in $Z'^l$
  • $Y''^l$: data labels in $\mathcal{C}_N^l$ for points in $Z''^l$
  • $s$: size of set $\mathcal{S}$

output:

  • $(\mathbf{w}_l, b_l)$: SVM model parameters

1: for $i = 1$ to $|\mathcal{C}_K^l|$ do
2:     compute $\theta_i^l$ from (3.4)
3: $\Psi_P^l = \{\}$ {build set of positive examples}
4: $\Omega \leftarrow$ all disjoint subsets of size $s$ and equal label in $Z''^l$
5: for all $\mathcal{S} \in \Omega$ do
6:     compute $\theta_{\mathcal{S}}^l$ from (3.3)
7:     compute $\hat{o} = o_{\mathcal{S}}^l$ from (3.1)
8:     $\Psi_P^l \leftarrow \Psi_P^l \cup \{[\theta_{\mathcal{S}}^l, \theta_{\hat{o}}^l]\}$
9: $\Psi_N^l = \{\}$ {build set of negative examples}
10: $\Omega \leftarrow$ all disjoint subsets of size $s$ and equal label in $Z'^l$
11: for all $\mathcal{S} \in \Omega$ do
12:     compute $\theta_{\mathcal{S}}^l$ from (3.3)
13:     compute $\hat{o} = o_{\mathcal{S}}^l$ from (3.1)
14:     $\Psi_N^l \leftarrow \Psi_N^l \cup \{[\theta_{\mathcal{S}}^l, \theta_{\hat{o}}^l\}$
15: $(\mathbf{w}_l, b_l) \leftarrow$binary-SVM.train($\Psi_P^l$, $\Psi_N^l$)
16: return $\mathbf{w}_l, b_l$
```

result of L classifiers $\{h_l(x)\}_{l=1}^L$. We will show next that for a sufficiently large number of classifiers L, there exists a threshold such that the probability of error of the final novelty classifier can get as close as we like to 0. In the following analysis we make 2 simplifying assumptions: (i) $|\mathcal{S}| = 1$; and (ii) the condition in line 3 in Alg. 3 is ignored, and lines 4-5 are executed $\forall l$.

We use the notations $\mathcal{C}, \mathcal{C}_K^l, \mathcal{C}_N^l$ defined in chapter 3.2, and let $\overline{\mathcal{C}}$ denote the set of all remaining classes not seen in the training set. We define the following L indicator random variables, which are not necessarily iid:

$$X_l(x) = \begin{cases} 1 & h_l(x) = 1 \\ 0 & h_l(x) = -1 \end{cases}$$

$$E(X_l(x)/y \in \overline{\mathcal{C}} \cup \mathcal{C}_N^l) = p_l, \quad E(X_l(x)/y \in \mathcal{C}_K^l) = q_l$$

Recall that classifier $h_l(x)$ is trained to return -1 when $y \in \mathcal{C}_K^l$ and 1 when $y \in \overline{\mathcal{C}} \cup \mathcal{C}_N^l$.

Successful training will therefore give us $p_l > q_l \geqslant 0 \; \forall l$; a weaker assumption is used here, as stated shortly.

Let $X(x) = \sum_{l=1}^{L} X_l(x)$ denote the random variable corresponding to the novelty score.

Training of a single novelty classifier h_l

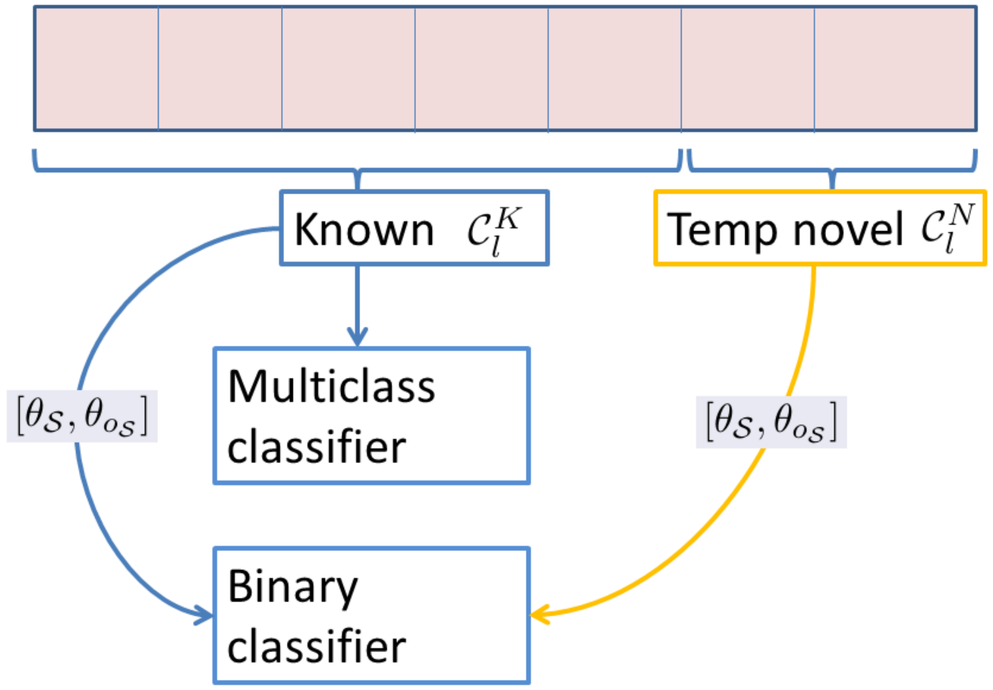
Known $\mathcal{C}_l^K$
Temp novel $\mathcal{C}_l^N$
Multiclass classifier
Binary classifier
$[\theta_{\mathcal{S}}, \theta_{o\mathcal{S}}]$
$[\theta_{\mathcal{S}}, \theta_{o\mathcal{S}}]$

Figure 3.1: The class labels in the training data are divided to 2 sets, $\mathcal{C}_N^l$ and $\mathcal{C}_K^l$. A multiclass classifier is trained using examples with labels in $\mathcal{C}_K^l$. Using this multiclass classifier, we compute a representation in $\mathcal{R}^2$, $(\theta_{\mathcal{S}}^l, \theta_{o\mathcal{S}}^l)$, for presumed known examples with labels in $\mathcal{C}_N^l$ and presumed novel examples with labels in $\mathcal{C}_K^l$. These representations are used to train a novelty-vs-known binary classifier.

Algorithm 3 Computing novelty score

```
input:
  • $\mathcal{S}$: set of test points known to share the same class label
  • $\mathcal{F}$: multiclass class classifier that was trained on the entire set
  • $\mathcal{F}^l$: multiclass class classifier that was trained on data from $\mathcal{C}_K^l$ classes
  • $\{\mathbf{w}_l, b_l\}$, $l \in [L]$, SVM parameters of novelty classifiers in ensemble $\mathcal{H}$
output:
  • $novelty\_score$
1: compute $\hat{o} = o_{\mathcal{S}}$ from (3.1) using $\mathcal{F}$
2: for $l = 1$ to $L$ do
3:     if $\hat{o} \in \mathcal{C}_K^l$ then
4:         compute $\mathbf{d}_l = [\theta_{\mathcal{S}}^l, \theta_{\hat{o}}^l]$ using $\mathcal{F}^l$ classifier, and(3.3),(3.4)
5:         $P_l(\mathcal{S}) = sign(<\mathbf{w}_l, \mathbf{d}> + b_l)$
6:     else
7:         $P_l(\mathcal{S}) = 0$
8: Return $\sum_{l=1}^{L}(P_l(S) == 1)$
```

We define

$$\begin{aligned}
\mu_{novel}^L =& E[X(x)/y \in \overline{\mathcal{C}}] = \sum_{l=1}^{L} p_l \\
\mu_{known}^L =& E[X(x)/y \in \mathcal{C}] = \sum_{l=1}^{L} \psi_{jl} \\
\psi_{jl} =& \begin{cases} q_l & y \in \mathcal{C}_K^l \text{ in partition l} \\ p_l & y \in \mathcal{C}_N^l \text{ in partition l} \end{cases}
\end{aligned}$$

Using the Chernoff bound[1], $\forall 0 < \delta < 1$ and $\forall L$

$$P[X(x) > (1+\delta)\mu_{known}^L / y \in \mathcal{C}] \leqslant \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \right)^{\mu_{known}^L} \tag{3.5}$$

$$P[X(x) < (1-\delta)\mu_{novel}^L / y \in \overline{\mathcal{C}}] \leqslant \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}} \right)^{\mu_{novel}^L} \tag{3.6}$$

Assuming that $p_l, q_l > 0 \; \forall l$, the probability of the events described in (3.5) and (3.6) gets sufficiently close to 0 for large enough L. See Chernoff bound illustration in Fig. 3.2

[1]We need to assume that $\{X_l(x)\}_{l=1}^{L}$ are conditionally independent given that y belongs to a class in either $\overline{\mathcal{C}}$ or $\mathcal{C}$, and that we can obtain L such conditionally independent classifiers for large L.

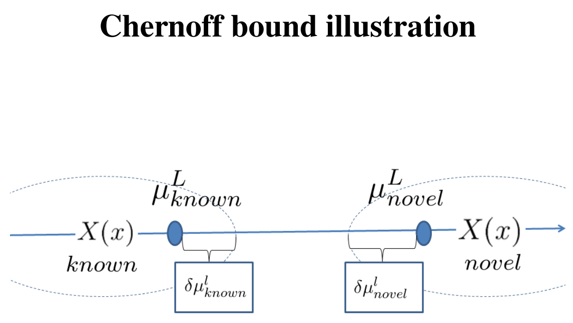

Figure 3.2: Given a large enough L, with high probability $X(x)$ when $y \in \overline{\mathcal{C}}$ is greater than $X(x)$ when $y \in \mathcal{C}$.

To show our main result, we need to make the following assumption whose implication will be discussed shortly - $\exists L_0 : [\mu^L_{novel} - \mu^L_{known}] \geqslant 2\delta_0 \ \forall L \geqslant L_0$. If we choose $\frac{\mu^L_{known}+\mu^L_{novel}}{2}$ as the novelty threshold, it follows from (3.5) and (3.6) and $\delta = \delta_0$ that there exists an L such that the probability of error can be small as desired.

Empirical look at classifier h_l The main assumption made in the discussion above requires that $\exists L_0 : [\mu^L_{novel} - \mu^L_{known}] \geqslant 2\delta_0 \ \forall L \geqslant L_0$. If $p_l > q_l \ \forall l$ (or 'almost' every l) including the limit of $l \to \infty$, and if $\frac{|\mathcal{C}^l_N|}{|\mathcal{C}|}$ is bounded by a preferably small number in $(0, 1) \ \forall l$, then this assumption eventually holds. This sufficient condition is still very weak: it requires that the training of each classifier h_l is successful in the sense that it identifies novel points as novel more often than it identifies known points as novel. This condition can be rewritten as the following two requirements:

R1. The distribution of the novelty measure $\theta_{\mathcal{S}}$ defined in (3.3) is different between the case when examples with the same label as $\mathcal{S}$ exist in the training set of the initial multiclass classifier, and the case when such points do not exist.

R2. The distribution of $\theta_{\mathcal{S}}$ when limited to classes in $\mathcal{C}^l_N$ is sufficiently similar to the distribution of $\theta_{\mathcal{S}}$ over classes in $\overline{\mathcal{C}}$.

We demonstrate the plausibility of these requirements with empirical evaluation as shown in Fig. 3.3. Here, the feature $\theta_{\mathcal{S}}$ indeed discriminates the behavior of novel from known examples, as required in R1. The plot also demonstrates that the distribution of $\theta_{\mathcal{S}}$ when uniformly sampling test points from $\mathcal{C}^l_N$ is similar to its distribution when uniformly sampling test points from $\overline{\mathcal{C}}$, as required in R2.

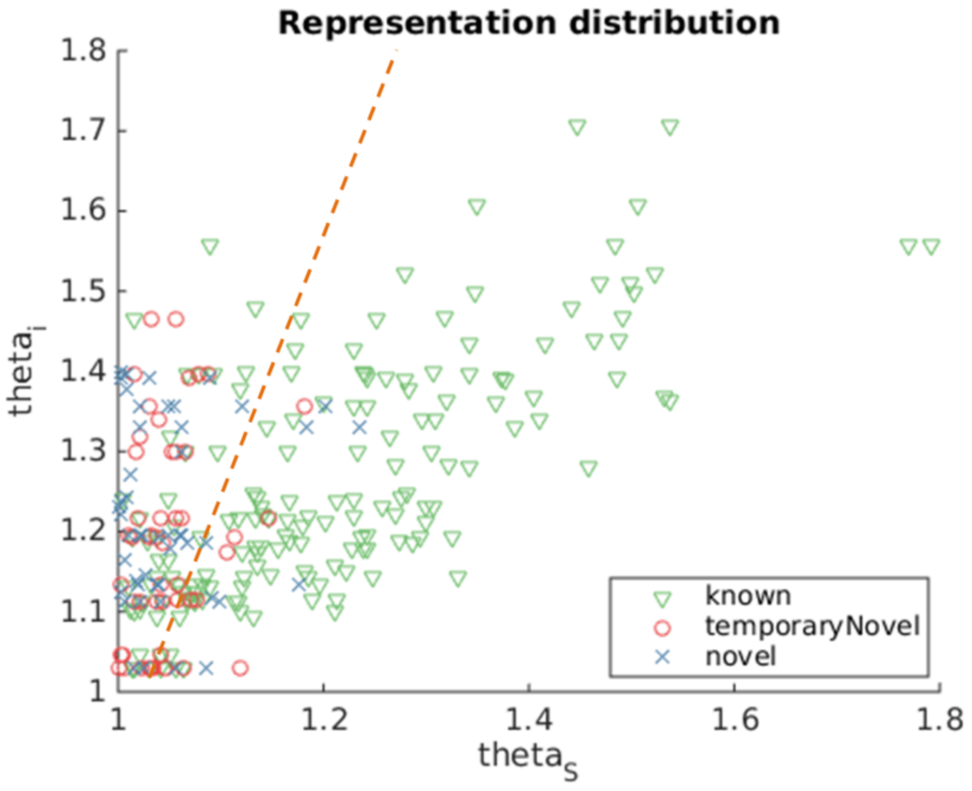

Figure 3.3: Empirical distribution of 3 types of test points in the space defined by the two features $\theta_{\mathcal{S}}$ and θ_i, including test points from $\overline{\mathcal{C}}$, $\mathcal{C}_N^l$, and from $\mathcal{C}_K^l$. We plot the line which separates $\overline{\mathcal{C}} \cup \mathcal{C}_N^l$ from $\mathcal{C}_K^l$. To generate this example, we used data from the Cifar-100 dataset.

3.5 Discussion

In our method, points are represented in the space of $[\theta_{\mathcal{S}},\ \theta_{o_{\mathcal{S}}}]$. As Fig. 3.3 demonstrates, in this representation novel points are separated from known points fairly robustly. Also, note that the linear separator in Fig. 3.3 is not parallel to the y-axis. This demonstrates the importance of using the class novelty score $\theta_{o_{\mathcal{S}}}$ to calibrate the point novelty score $\theta_{\mathcal{S}}$. Finally, note that while novel points are separated from known points fairly reliably, we should expect many errors when using a single classifier h_l as in Fig. 3.3. To remedy, our algorithm uses an ensemble of such binary classifiers, which contributes to its good novelty detection performance.

Chapter 4

Experimental Evaluation

Since the focus of this work is novelty detection when some class labels are missing from the training set, we used in our evaluation two large datasets with many class labels and many examples per class. Specifically, we used the Caltech-256 [11] and Cifar-100 [14] datasets. Experiments are described respectively in Section 4.2 and Section 4.3. We compared our method to a number of representative novelty detection or reject methods, as described in Section 4.1. Almost all existing novelty detection methods are designed to classify one example, which in our formulation implies that $\mathcal{S}$ is a singleton. We therefore varied the value of the set size $|\mathcal{S}|$ to investigate how the different algorithms take advantage of the availability of a larger set size.

Since novelty detection methods typically define a *novelty score* to be compared against a threshold, the performance of the different methods depends on the value of the threshold. As the threshold is increased, the method would detect more true novel events while at the same exhibit higher false positive rates. To evaluate these methods we therefore use as customary the Receiver Operating Characteristic (ROC) curve, where for each method the *novelty score* is compared to a varying threshold. To obtain a single measure of success, we use (as customary) the Equal Error Rate (EER) and the Area Under Curve (AUC).

The experiments were conducted in a 10-fold cross-validation manner, where in each experiment 10% of the class labels in the training data are set aside (as novel classes) to test the method, and the rest of the training data is used for actual training. This design was repeated $3-4$ times with a different set of labels set aside as novel, and the results are plotted in the graphs below (average and standard deviation over different experiments for each dataset separately).

4.1 Methods used for comparison

As stated above and explained in the introduction, we chose a few representative (and simple) novelty methods to compare our method against. Our choice of methods was motivated by the desire to represent the different kind of approaches as reviewed in the introduction, and specifically by the empirical observation made in a recent review article on novelty detection [8] which identified the approach based on k-nearest-neighbor (k-NN) as the most effective novelty detector in their evaluation.

The first method we used for comparison, which belongs to the same discriminative framework (as opposed to generative) as we follow here, is the one-class SVM (OCSVM) [21] - a popular novelty detection algorithm. OCSVM uses the kernel trick to construct a hyperplane that separates the normal data from the origin with maximum margin in feature space. We used all the datapoints from the known classes in the training set to compute the discriminating hyperplane using the Gaussian kernel. As is customary, given a new point $\mathbf{x}$ we used the margin value multiplied by -1 as the *novelty score*. When $|\mathcal{S}| > 1$ we used the mean margin value as the *novelty score* of set $\mathcal{S}$. For implementation we used the publicly available code in python scikit-learn 'One-class SVM with non-linear kernel'.

The choice of the second representative method is motivated by the empirical observations reported in [8]. This study performed a comparative evaluation of four widely used methods. The experimental results showed that the k-NN novelty detection method exhibits competitive overall performance when compared to other methods in terms of AUC. Therefore we compared our method to k-NN. We implemented the k-NN method according to the implementation described in [8] and detailed in chapter 2. As in [8], the euclidean distance is used. For $k > 1$, we used the average distance. We ran this method using $k = \{1, 2, 5\}$.

Finally, we compared our method to another method that is based on applying some evaluation operator to a vector of soft label as reviewed in [16]. [16] emphasized the distinction between two separate types of novelty - events that arise from an unknown class (*distance reject*), and events that fit several classes (*ambiguous reject*). Different measures applied to the vector of soft labels were reviewed or defined, but focusing our interest on *distance reject* as we do here, all the operators essentially performed the same thing, applying *argmax* to the soft labels vector. Accordingly, we used max confidence multiplied by -1 as the *novelty score* representing this family of measures. When $|\mathcal{S}| > 1$, we used the average vector of soft labels taken over all elements of $\mathcal{S}$, as in (3.1).

In addition, we computed a *novelty score* based on our own method stripped of its learning session, using only θ_S, without training a classifier which compares it to θ_i. This method is denoted in the graphs below as 'threshold'. We also included in our comparisons the recent KNFST method (described in the introduction) for which we used the code from the authors' website.

4.2 Caltech-256 dataset

The Caltech-256 dataset [11] contains 256 classes with at least 80 examples per class. 6 classes were excluded from the study due to their similarity to other classes in the dataset, leaving us with a total of 250 classes.

Training Details From each class, 60 randomly chosen images were used to train the initial CNN-based multiclass classifier, 10 were used to train a binary classifier h_l, and the remaining images were used for testing. For each image that was used to train the multiclass classifier we also added the horizontally mirrored image. To make the time-consuming step of training a multi-class classifier simple and efficient, we used the pre-trained CNN *Overfeat* [22]. In this we adopted the procedure proposed in [19], where it was shown that this pre-trained network provides a very effective representation which can be followed by any multiclass classifier. Specifically, the representation that is used is taken from the 19th layer, resulting in a $d = 4096$ feature vector. Given this baseline 4096 feature vector representation for each image, we trained a flat network constructed from 2 affine layers (with an intermediate activation layer) to classify all the classes in the training set. The output of the network provided us with a soft label representation for each image, a vector in $\mathcal{R}^{k'}$ where $k' = 224$ is the number of classes in the training set.

A 10-fold cross-validation procedure was repeated 4 times, each time dividing the 250 classes into different sets of 224 known and 26 novel classes. In each of the 4 repetitions we trained 40 binary classifiers, each of which was trained to discriminated between presumed-known and presumed-novel classes, using a different partition of the set of 224 known classes to 198 presumed-known classes and 26 presumed-novel classes. We varied the size of set $\mathcal{S}$ in the range $1 - 5$. Two different image representations were used in the evaluation of the k-NN and one-class SVM methods. The first used the vector of confidence values in $\mathcal{R}^{224}$ as described above. The second used the original 4096 feature vector representation, followed by PCA-based dimensionality reduction to $\mathcal{R}^{250}$ (performance was incredibly poor without this added dimensionality reduction step). When running the KN-FST code we used the original 4096 feature vector representation with polynomial kernel, whose degree was optimized to match the results reported in [4] for 5 and 10 known classes. Selected results are shown in Fig. 4.1, while all results are shown in Tables 4.1,4.2.

4.3 Experiments on Cifar-100

Cifar-100 [14] consists of 32×32 color images belonging to 100 classes. There are 600 examples from each class; we used 500 to train a CNN-based multiclass classifier, 50 to generate sets to train the binary classifiers h_l, and the remaining images were used for testing.

Training Details Network In Network (NIN) [17] was used as the multiclass classifier. Images were pre-processed by global contrast normalization and ZCA whitening as in [17]. An 11-fold cross-validation procedure was repeated 3 times, each time dividing the 100 classes into different sets of 89 known and 11 unknown classes. In each of the 3 repetitions we trained 27 binary classifiers, each of which was trained to discriminate between presumed-known and presumed-novel classes, using a different partition of the set of 89 known classes to 78 presumed-known classes and 11 presumed-novel classes. We varied

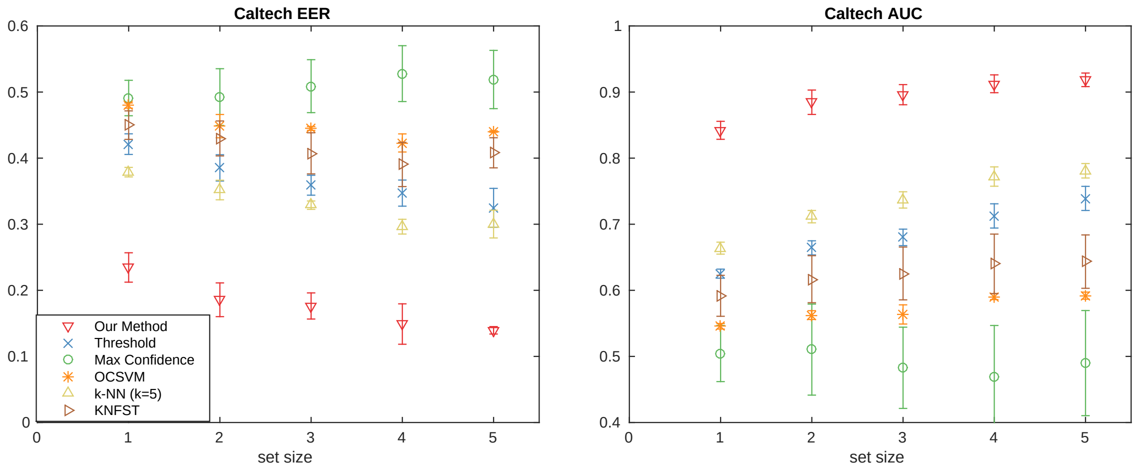

Figure 4.1: The EER (left) and AUC (right) of the different novelty scores *based on the corresponding ROC curve of each method on test data using the Caltech-256 dataset. (See text for explanation of methods used for comparison.) When evaluating the one-class SVM (OCSVM) and k-NN methods, we show results when using the "original" feature vectors as input, since these gave much better results as compared to using the confidence values.*

Table 4.1: Novelty Detection EER and AUC (in percent) for the Cifar-100 and Caltech-256 datasets given set $\mathcal{S}$ of size 1.

	EER		AUC	
Method	Cifar-100	Caltech-256	Cifar-100	Caltech-256
Our method	0.37± 0.05	0.26± 0.04	68± 3	79± 2
Simple threshold	0.40± 0.01	0.42± 0.02	65± 1	62± 1
Max confidence	0.40± 0.04	0.46± 0.02	63± 5	57± 1
OCSVM	0.50± 0.01	0.48± 0.01	49± 1	52± 1
k-NN (k=1)	0.47± 0.01	0.47± 0.01	54± 2	54± 1
k-NN (k=5)	0.47± 0.03	0.48± 0.01	55± 3	60± 1
OCSVM on original	NA	0.48± 0.004	NA	54± 1
k-NN on original(k=1)	0.50±0.01	0.42± 0.01	51±2	62± 1
k-NN on original(k=5)	0.50±0.01	0.38± 0.01	50±2	66± 1
KNFST	NA	0.44± 0.02	NA	59± 3

the size of set $\mathcal{S}$ in the range $1-5$. Two different image representations were used in the evaluation of the k-NN and one-class SVM methods. The first used the vector of confidence values in $\mathcal{R}^{89}$ as described above. The second used Histogram of Oriented Gradients (HOG) representation in $\mathcal{R}^{324}$ followed by PCA dimensionality reduction to $\mathcal{R}^{100}$. Selected results are shown in Fig. 4.2, while all results are listed in Tables 4.1,4.2.

Table 4.2: Novelty Detection EER and AUC (in percent) on Cifar-100 and Caltech-256 given set $\mathcal{S}$ of size 5.

	EER		AUC	
METHOD	CIFAR-100	CALTECH-256	CIFAR-100	CALTECH-256
OUR METHOD	0.11± 0.02	0.10± 0.03	93± 2	96± 2
SIMPLE THRESHOLD	0.17± 0.01	0.32± 0.03	90± 1	73± 2
MAX CONFIDENCE	0.30± 0.08	0.39± 0.04	75± 8	63± 3
OCSVM	0.53± 0.01	0.48± 0.01	49± 1	44± 2
k-NN (K=1)	0.43± 0.03	0.44± 0.03	59± 4	54± 3
k-NN (K=5)	0.44± 0.06	0.37± 0.02	59± 7	60± 2
OCSVM ON ORIGINAL	NA	0.44± 0.02	NA	58± 1
k-NN ON ORIGINAL(K=1)	0.48±0.03	0.34± 0.02	54±4	72± 1
k-NN ON ORIGINAL(K=5)	0.47±0.04	0.30± 0.02	52±4	78± 1
KNFST	NA	0.40± 0.02	NA	64± 4

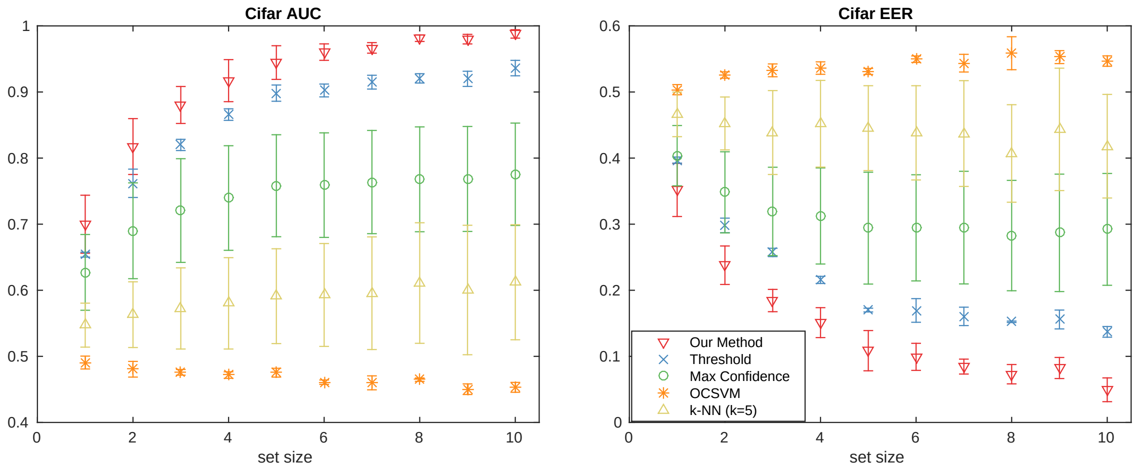

Figure 4.2: The EER (left) and AUC (right) of the different novelty scores *based on the corresponding ROC curve of each method on test data using the Cifar-100 dataset.*

4.4 Discussion

The results above demonstrate very clearly the advantage of our proposed novelty score as compared to the standard (and widely used) alternative methods. With the Caltech-256 dataset, using θ_S as novelty score had comparable performance to other methods, while our proposed score achieved much better performance than all alternative methods. With the Cifar-100 dataset, simply using θ_S as novelty score gave better results than other methods (see Fig. 4.2), with some additional improvement obtained when using the final novelty score based on learning. The fact that using θ_S as novelty score gave much better results

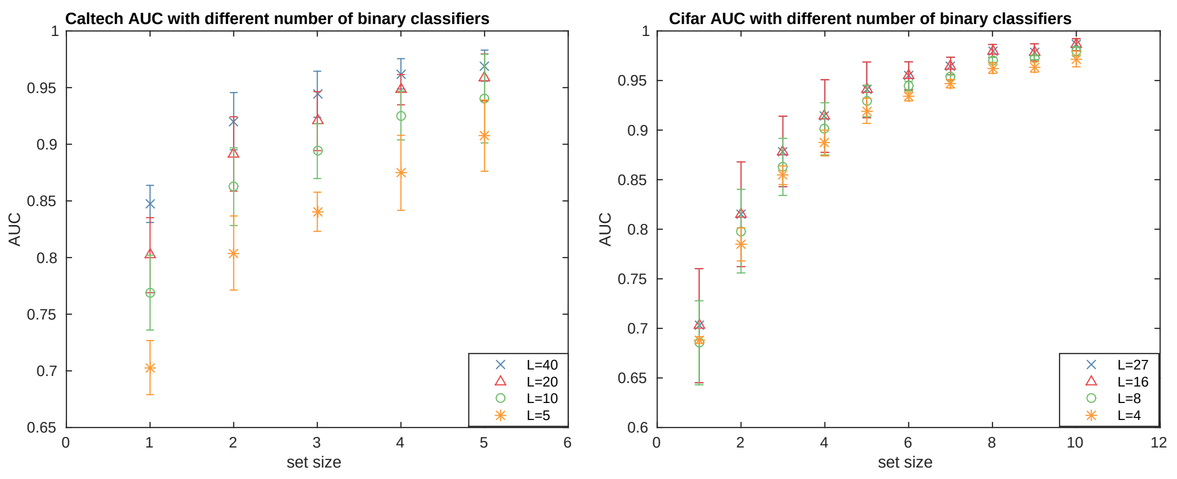

Figure 4.3: The AUC of novelty score *based on the corresponding ROC curve of our method, with different number of binary classifiers (L), using the Caltech-256 dataset (left) and cifar-100 dataset (right).*

than other methods, on Cifar-100 dataset but not on Caltech-256 dataset, may be due to a bit different multiclass classifier training procedure. At the first step we train a multiclass classifier. For Cifar-100 dataest we train NIN, a CNN that was designed as Cifar-100 multiclass classifier, while for Caltech-256 we use pre-trained *Overfeat* CNN last layer representation and train a flat net as a multiclass classifier. Therefore, NIN soft label vectors might be much more informative, and this enables more accurate novelty detection using simply θ_S as novelty score.

Since k-NN and OCSVM were initially developed to work directly with feature vector representations rather than vectors of confidence values, we ran these methods also using the ”original” feature vector representations. Given the Caltec-256 dataset, we used the original representation in $\mathcal{R}^{4096}$ followed by PCA down to $\mathcal{R}^{250}$. The performance of k-NN was improved when used in this feature space, as seen in Tables 4.1-4.2: with this representation k-NN works relatively well, but not as well as our learning method. Given the Cifar-100 dataset, we used for original representation the Histogram of Oriented Gradients (HOG) representation in $\mathcal{R}^{324}$ followed by PCA down to $\mathcal{R}^{100}$. This representation did not improve the performance of k-NN, while OCSVM failed completely.

To achieve high resolution in the novelty score, we used an ensemble of $L = 40$ binary classifiers when using the Caltech-256 dataset, and $L = 27$ binary classifiers when using the Cifar-100 dataset. To investigate the dependence of our method on this parameter, we checked the method’s performance using fewer classifiers, observing similar results with fewer classifiers, and a very slow overall degradation of the results down to $L = 5$ binary classifiers, as appears in Fig 4.3. For Cifar-100 dataset the degradation is slower, probably due to the same reason that cause simply using θ_S as novelty score, achieve pretty good results. We also note that the relative advantage of our method over other methods increases as we increase the set size $\mathcal{S}$.

When comparing our results to a recent comparative evaluation [8], we see that when comparing the performance of the standard alternatives methods tested there - max likelihood (similar to max confidence when confidence is normalized to represent probabilities), k-NN and SVDD [23] (strongly related to OCSVM), their reported relative ordering is similar to what we see above. Our experiments still provide added value because of the larger datasets used here: whole images as compared to $2-57$ features in [8], hundreds of classes vs. a few to a few dozens, and many more examples per class.

Comparing our method to the KNFST (on Caltech-256 data base) shows that in the framework where there are 224 known classes, our method achieves much better performance. KNFST gives good results when 5-20 classes are known. In the complicated cases when considerably more classes are known, KNFST method demonstrates pretty poor results, compared to our method.

Chapter 5

Summary and Discussion

In this work we address the problem of classification with insufficient information about the set of class labels, and specifically when some class labels are not represented in the training set. This may happen because some class labels are rare and may be missed during training, or since new classes that were not exist during training stage, may appear later. For example, creation of a new species of bacteria or virus, appearance of a face of a new person, or appearance of a completely new object due to some invention. We describe a method for the detection of novel classes, and specifically for the computation of a *novelty score* for each test example based on prior processing where the assignment of this example to each known class is evaluated in a soft manner. This can be accomplished, for example, from the output of any discriminative multiclass classifier, or a generative model of the data.

Our method starts out by comparing the two top confidence values in the vector of soft assignments, a measure which is subtly different from what has been used previously. However, what distinguishes our method most is the training of an ensemble of binary novelty classifiers, which are trained to distinguish known from unknown classes based on the aforementioned measure and some random partition of the training data into presumed-known and presumed-novel sets. The final *novelty score* is computed based on the output of this ensemble of binary classifiers.

We present two more innovative ideas. The first one, is the learning of the behavior of data points from novel classes. Unlike most of novelty detection algorithms where the main idea is to find good boundaries of the known classes data space, our method learns the representation of a sample from a novel class, in the context of a multiclass classifier that was trained only on known classes. The representation that our method learns is a two-parameter representation that we define, as appears in 3.1. This representation enables calibrating the new sample according to its predicted class. The second innovative idea is to investigate the case where one knows that a group of test points belongs to the same (known or novel) class. When the group size is one this scenario reduces to the usual novelty detection framework, but when group size is greater than one, this produces prior knowledge. This is relevant to sequential frames from a movie, collecting group of samples from a bacteria colony, and many others. This additional prior knowledge enables much better novelty detection performance, thus it should be used when it is possible.

We tested our method in a comparative framework meant to evaluate the ability of each method to detect novelty in the kind of scenario addressed here, when some class labels are not available during training. Testing novelty detection with two relatively large datasets of images, our final *novelty score* is shown to perform much better than three other standard alternative methods that are commonly used, and that have been shown to be rather effective in previous comparative studies. The marked advantage of our method can be attributed to the learning step inherent in our method, and to the fact that our method is designed to detect novel class labels, while other methods may be more suitable for scenarios where novel points are due to anomalies in the data or outliers.

Further work should expand this method to enable two kinds of reject- *distance reject* and *ambiguity reject*, as defined in [9]. *Distance reject* for samples which belong to novel classes, and *ambiguity reject* for samples that might fit several classes. Adding the concept of *ambiguity reject* to our method, may improve the signal to noise ratio of novelty detection where only samples with *distance reject* should be viewed as novel, and improve classification accuracy by classifying only samples that are not rejected.

Bibliography

[1] C. M. Bishop. Novelty detection and neural network validation. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 141, pages 217–222. IET, 1994.

[2] C. M. Bishop. Pattern recognition. *Machine Learning*, 2006.

[3] P. Bodesheim, A. Freytag, E. Rodner, and J. Denzler. Local novelty detection in multi-class recognition problems. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 813–820. IEEE, 2015.

[4] P. Bodesheim, A. Freytag, E. Rodner, M. Kemmler, and J. Denzler. Kernel null space methods for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3374–3381, 2013.

[5] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[6] C. K. Chow. An optimum character recognition system using decision functions. *Electronic Computers, IRE Transactions on*, (4):247–254, 1957.

[7] C. K. Chow. On optimum recognition error and reject tradeoff. *Information Theory, IEEE Transactions on*, 16(1):41–46, 1970.

[8] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire. An experimental evaluation of novelty detection methods. *Neurocomputing*, 135:313–327, 2014.

[9] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern recognition*, 26(1):155–165, 1993.

[10] C. Frélicot and H. Le Capitaine. *Class-selective rejection rules based on the aggregation of pattern soft labels*. INTECH Open Access Publisher, 2010.

[11] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. *Technical Report*, 2007.

[12] F. E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[13] V. Hautamäki, I. Kärkkäinen, and P. Fränti. Outlier detection using k-nearest neighbour graph. In *ICPR (3)*, pages 430–433, 2004.

[14] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] H. Le Capitaine and C. Frelicot. A family of measures for best top-n class-selective decision rules. *Pattern Recognition*, 45(1):552–562, 2012.

[17] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[18] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[19] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[20] T. Scheffer, C. Decomain, and S. Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[21] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *NIPS*, volume 12, pages 582–588, 1999.

[22] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[23] D. M. Tax and R. P. Duin. Support vector domain description. *Pattern recognition letters*, 20(11):1191–1199, 1999.

[24] D. Yu, G. Sheikholeslami, and A. Zhang. Findout: finding outliers in very large datasets. *Knowledge and Information Systems*, 4(4):387–412, 2002.