Multi-class Novelty Detection Using Mix-up Technique

Supritam Bhattacharjee
Indian Institute of Science
supritamb@iisc.ac.in

Devraj Mandal
Indian Institute of Science
devrajm@iisc.ac.in

Soma Biswas
Indian Institute of Science
somabiswas@iisc.ac.in

Abstract

Multi-class novelty detection is increasingly becoming an important area of research due to the continuous increase in the number of object categories. It tries to answer the pertinent question: given a test sample, should we even try to classify it? We propose a novel solution using the concept of mix-up technique for novelty detection, termed as Segregation Network. During training, a pair of examples are selected from the training data and an interpolated data point using their convex combination is constructed. We develop a suitable loss function to train our model to predict its constituent classes. During testing, each input query is combined with the known class prototypes to generate mixed samples which are then passed through the trained network. Our model which is trained to reveal the constituent classes can then be used to determine whether the sample is novel or not. The intuition is that if a query comes from a known class and is mixed with the set of known class prototypes, then the prediction of the trained model for the correct class should be high. In contrast, for a query from a novel class, the predictions for all the known classes should be low. The proposed model is trained using only the available known class data and does not need access to any auxiliary dataset or attributes. Extensive experiments on two benchmark datasets, namely Caltech 256 and Stanford Dogs and comparisons with the state-of-the-art algorithms justifies the usefulness of our approach.

1. Introduction

Deep learning approaches have achieved impressive performance for object recognition and classification task [22] [39] by using large networks trained with millions of data examples. However, these networks usually work under a closed set assumption, and thus try to classify each query sample, even if it does not belong to one of the training classes. For example, a neural network classifier trained to classify fruits, might classify an input from a completely different category, say "bird" into one of the fruit classes with high confidence, which is unlikely to happen if a human does the same task. To make the systems more intelligent and better suited to real-world applications [34] [26], they should be able to understand whether the input belongs to one of the trained classes before trying to classify it.

This problem is addressed in recent literature as out-of-distribution detection [11], anomaly detection [46], novelty detection [30] [1] and one-class classification [31] [36], each having subtle differences between them. One class classification rejects all the classes as outliers except the concerned class. In out-of-distribution paradigm, the algorithm determines whether samples are coming from other data-sets or distribution. Such algorithms often require the availability or prior knowledge about the out-of-distribution data for proper functioning.

In this work, we address the multi-class novelty detection task [32], where given a query, the goal is to understand whether it belongs to one of the training classes. This is very challenging, since the novel data can come from the same data distribution as that of the training data. Here, we propose a novel framework, termed *Segregation Network*, which utilizes the *mix-up* technique [42] for this task. The network takes as input a pair of data points, and a third interpolated data point which is generated by mixing them together using a variable ratio. The goal is to determine the constituent classes and the respective proportions by which the two inputs have been mixed to generate the interpolated data point. To this end, we design a novel loss function called Constituency loss for our objective. Once the network is trained, given an unknown query, we mix it with the known class prototypes in a predefined proportion and pass it through the network. Based on the network output, we infer whether the query belongs to the known set of classes or a novel class unknown to the system. The main contributions of the proposed approach are as follows:

- We propose a novelty detection framework, termed as *Segregation Network*, which uses the mix-up technique, to detect whether the test query belongs to one of the known classes or to a novel class.

- We design a novel loss function called Constituency loss to train the proposed Segregation Network.

- The proposed network can be trained only using the available training data and does not require access to any auxiliary or out-of-distribution dataset as done in similar methods like [32]. This is advantageous as the collection of auxiliary data is often difficult, expensive and might be data dependent with respect to the known class set.
- During testing, the proposed network compares the unknown query with the set of known class prototypes. We also develop an efficient version of our algorithm, without any significant drop in performance, which utilizes the softmax confidence outputs of the pre-trained network.
- We perform experiments on two standard benchmark datasets for novelty detection and the results obtained compare favorably with the state-of-the-art method which leverages auxiliary training data.

The rest of the paper is organized as follows. A brief description of the related work in literature is provided in Section 2. The proposed approach is discussed in Section 4 and the experimental evaluation is described in Section 5. The paper ends with a brief discussion and conclusion.

2. Related Work

The foundation of this work is based on two threads of machine learning research, namely novelty detection algorithm and mix-up based learning techniques.

Novelty Detection: This problem is an active area of research for detecting outliers in data. There has been both statistical [40] [44] [20] [13], distance based [21] [18] [14] and deep learning based approaches [33] [2] [23]. Statistical methods generally focus on trying to fit the distribution of the known data using probability models [9]. Distance based algorithms usually use some transforms and then identify novel classes by thresholding the distance with known examples. The assumption is that the known class examples will be much closer to the known class representatives than the unknown ones in the transformed space. A relatively recent work in this direction is Kernel-Null Foley-Sammon Transform (KNFST) [7] for multi-class novelty detection. Here, the same class points are projected into a single point in the null space, and during testing, the distance with respect to the class representative is thresholded to get a novelty score. This algorithm was improved to handle incremental incoming classes and subsequently update its novelty detector in [25]. Deep learning based approaches such as Open-max [3] fits a Weibull-distribution to determine the novelty. The generative version of this approach was proposed in [15], where unknown samples were generated. Several one-class deep learning based novelty detection have been proposed in recent literature [35] [31] [36]. The work in [32] designs a novel training paradigm where a reference set is used to learn a set of negative filters that will not be activated for the known category data. To this end, they design a novel loss function called membership loss. Masana *et al.* [27] propose a method to improve the feature space by forming discriminative features with contrastive loss for this task. Out-of-Distribution detection algorithms [11] [24] [43] also addresses a similar task, having subtle difference with the problem addressed in this work. These approaches assume that the *novel* or out of distribution data is strictly outside the data manifold on which the base network is trained. Thus, novelty detection is in general more challenging than out of distribution detection [32].

Mixing: Learning algorithms involving interpolation or mix-up between classes has been recently introduced in the community. In one of the early works, Bengio *et al.* [4] employed *mixing* technique to better understand the underlying factor of disentanglement in data. Recently, [42] proposed to improve the classification task by interpolating between classes. In [41], sound recognition is done by feeding data between class (or mixed up) samples to improve the performance. Another work along similar lines is proposed in [45]. While the mentioned works interpolate in the input space, [5] interpolated in the latent space of auto-encoders to generate images with smooth transition from one class to another using adversarial regularizer. Several other works try to interpolate or mix data from different classes in the the latent space of auto-encoders [12] [28] [17] [8] [29] for different purposes and using various methods. In contrast, in our work, we interpolate in the feature space to train our model.

3. Motivation

The basic idea behind the development of the proposed algorithm is provided in Figure 1, where we highlight the training (top row) and testing (bottom row) strategies. The different color schemes indicate the different categories of data. The bi-directional arrow indicates that samples from these two categories are being used to generate the interpolated data point. The desired prediction of the trained model is shown in a color-coded bin.

During training, the proposed network aims to segregate the interpolated point with respect to the two inputs. For the example shown in Figure 1(a, b), the interpolated point lies close to the orange class, since higher mixing coefficient has been used for that class. So the model should ideally predict the constituent classes (as shown in the color coded bins) with higher weightage to the orange class. In Figure 1(c), the interpolated point should be predicted to belong to only the orange class, as two samples from the orange class

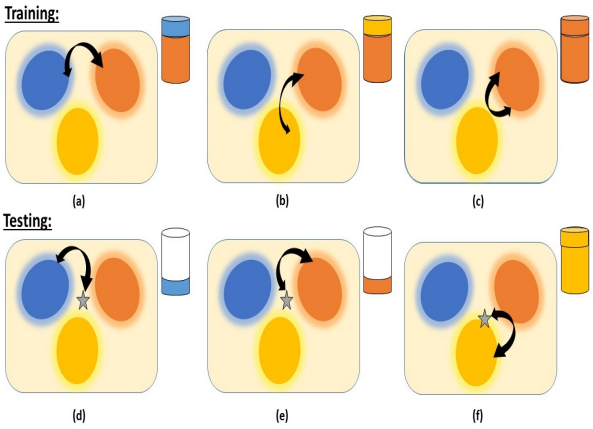

Figure 1. Illustration showing the motivation behind our proposed approach during the training and testing stage.

have been used to generate the interpolated point.

During testing (as shown in bottom row), we generate the interpolated point using the query (denoted as gray star) and the class prototypes of the known classes, and try to predict the constituent classes with respect to the two input points. We give a larger weightage to the query while generating the interpolated point. In Figure 1(d,e), since the query does not belong to any of the known class set, the prediction of the network should consist of only the known class mixing coefficient (shown in the color bins). In case the query comes from the known set (yellow class in Figure 1(f)), the prediction of our network should ideally reflect that with a high value in the color coded bin. Thus the output of the network can be used to determine whether the input belongs to one of the known classes or to a novel class.

4. Proposed Method

In this section, we describe the network architecture of the proposed Segregation Network(SN), the novel loss function used to train the model and the training and testing protocol. First, we describe the notations used.

Notations: Let the input data be represented as $\mathbf{X}^{tr} \in \mathbb{R}^{d_t \times N}$, N being the number of training samples and d_t being its feature dimension. Let the one-hot labels be denoted as $\mathbf{L}^{tr} \in \mathbb{R}^{K \times N}$, where K is the number of training or known classes. We define the known class set to be $C^s = \{C_1, C_2, ..., C_K\}$, and thus $|C^s| = K$. In the general scenario, the testing data can come from the seen classes or from unseen/novel classes, for which no information is available to the system. During testing, given a query, the goal is to determine whether it comes from set C^s or not, i.e. whether it belongs to a seen class or a novel class. Classifying the known examples into its correct class is not the focus of this work and can be done using the base classifier trained using the training data.

Base Classifier and Features: Given the training data, a base classifier is trained to classify an input query into one of the given classes. Since the classifier is now required to work in a general setting, the input can potentially come from an unseen class, and thus should not be classified by the classifier into one of the seen classes. Given an input, the novelty detector is designed to take the output features of the base network (before they are classified) and decide whether the input belongs to one of the seen classes or to a novel class. If it belongs to a seen class, it can be sent back to the classifier for classification. The proposed novelty detection framework is general and can work with any classifier model. In this work, we use pre-trained AlexNet [22] and VGG16 [39] architecture as the base classifier. These networks are fine-tuned on the respective known class datasets and the extracted features are normalized and given as input to our network. Now, we describe the details of the Segregation Network.

4.1. Segregation Network

The proposed network consists of three fully connected (fc) layers with ReLU activations and dropout between each layer except the final fc layer. The final layer is of dimension K (number of seen classes). Sigmoid is used as the final layer activation function as the output of Sigmoid is between $[0, 1]$, which can be interpreted as the proportion of the mixing classes in our case. In our design, the network has a $512-1536-256$ architecture, with the numbers denoting the length of each fc layer. We use the Adam optimizer with a learning rate of 0.001 for training our model. An illustration of the proposed network is shown in Figure 2.

The network takes as input a triplet set of data samples $\{x_i, x_j, x_k\}$, where x_i, x_j are data from the training set and x_k is the mixture obtained by mixing x_i and x_j in some proportion. Let us denote the output of the first fc layer, which is shared by all three inputs, as $\{x_i^1, x_j^1, x_k^1\}$. Then $\{x_i^1, x_j^1, x_k^1\}$ is concatenated together to form x_{ijk}^1 which is then passed forward through the rest of the network. In our implementation, we have used features from the fine-tuned AlexNet [22] or VGG16 [39] deep networks, which are of very high dimension. Thus, the first fc layer serves as a dimensionality reduction layer, the output of which is then concatenated and passed through the rest of the network structure. The final output of the network after passing through the Sigmoid activation function is denoted as u.

Training the model : The network is trained such that given an interpolated input, it will decouple/segregate the input data into its constituents. This property is exploited in the following way. Given a pair of feature vectors $\{x_i, x_j\}$, we perform convex combination on this pair to produce x_k , where $x_k = \alpha x_i + (1 - \alpha)x_j$, $\alpha \in (0, 1)$. We

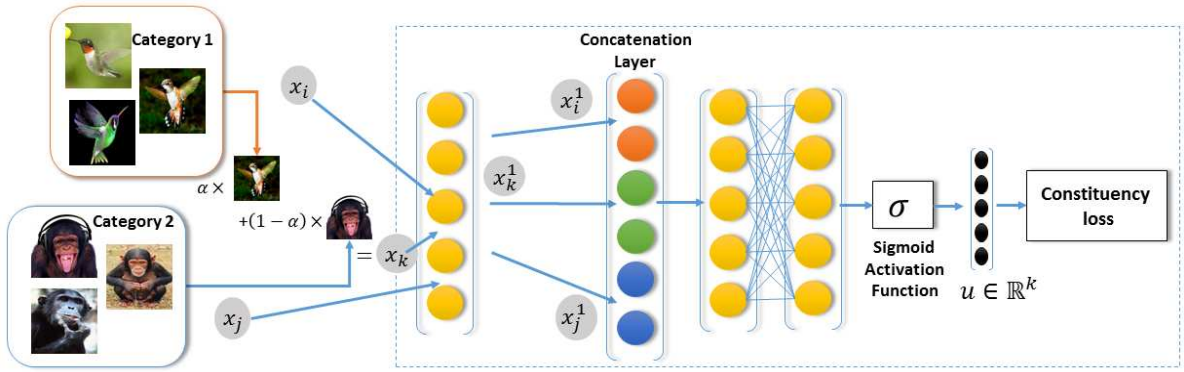

Figure 2. Illustration of the proposed network. The network accepts feature vectors x_i belonging to C_1 (of "Birds" category) and x_j belonging to C_2 (of "Chimpanzee" category) to create hybrid data in the feature space, x_k. All these three vectors $< x_i, x_j, x_k >$ are first passed through the first fully connected layer of the network to be transformed into a lower dimensional vector before being concatenated together to pass it through the rest of the network. The final activation layer is kept Sigmoid so that the output of network u can be used to predict the mixture ratio. We train the model using our proposed novel Constituency loss function.

feed these three feature vectors $\{x_i, x_j, x_k\}$ to the network. The output of the network is a K dimensional vector from the final Sigmoid layer $u = [0,1]^K$. Since the output is passed through the Sigmoid activation function, each element of the K-dimensional vector is bounded between $[0, 1]$. In addition, each element denotes the proportion by which the mixed sample x_k has been constructed from that training class. For example, an output of $[0, 0.6, 0.4, 0, 0]$ (i.e. there are five known classes) indicates that the mixed sample x_k has been constructed as $x_k = 0.6x_i + 0.4x_j$ where $x_i \in C_2$ and $x_j \in C_3$. Given $x_k = \alpha x_i + (1-\alpha)x_j$, the following cases may arise,

- If, $x_i \in C_p$ and $x_j \in C_q$, where $p \neq q$, and both $(C_p, C_q) \in C^s$ i.e., belongs to the seen classes set, we should get the output of the model such that, $u[p] = \alpha$ and $u[q] = 1 - \alpha$, while $u[r] = 0$ for $r \neq \{p, q\}$. We consider such a pair to be a non-matched pair as the interpolated point x_k lies somewhere in between the two classes based on the value of α.

- If, both $x_i, x_j \in C_p$, $C_p \in C^s$, the network should ideally output $u[p] = 1$ and $u[r] = 0$ for $r \neq p$. This is because a mixed element constructed from two data items of the same class should ideally belong to the same class also. We consider such a pair to be a matched pair.

- During testing in general scenario, we pair the query sample with class prototypes of the known classes, and so a third case may arise if the query belongs to a novel class. Here, since one of the two inputs to the network is seen, only the output node corresponding to that class should be non-zero and equal to the proportion of this class in the generated mixture. We do not explicitly train the network for this scenario, since we do not assume any auxiliary dataset. So, we consider only the first two cases for training.

Algorithm 1 Algorithm for training the Segregation Network

1: **Input** : $\mathbf{X}^{tr}$ is the input data with their provided labels $\mathbf{L}^{tr} \in C^s$.
2: **Output** : Trained Segregation Network model to detect novel samples.
3: **Initialize** : Initialize the network parameters of Segregation Network. Extract the fine-tuned features for $\mathbf{X}^{tr}$ using AlexNet [22] or VGG16 [39].
4: Randomly generate the mixing coefficient value α.
5: Randomly take pairs of training data $x_i, x_j \in \mathbf{X}^{tr}$ to construct x_k using the mixing coefficient α
6: Feed-forward the triplet data pair (x_i, x_j, x_k) through the network.
7: Compute the constituency loss $\mathcal{L}^{cons}$ and back-propagate it back to train the network.

Note that as the final activation function is the Sigmoid layer and not the standard softmax, total sum of u may not be equal to 1. This is important, since if the input belongs to a novel class, the network will only consider the mixing proportion of the known class. So the proportion of unknown class in the mixture will be ignored and thus the sum will not be equal to 1.

Our network needs to be trained in such a way that the value of u peaks at the position of the constituent classes and also gives the proper mixing proportions. Since, we do not have the softmax output, we cannot use cross-entropy loss function to train this model. In addition, cross-entropy loss function tries to maximize the probability of a sample to belong to a particular class which goes against our

desired objective, where a given example can come from outside the known class set. Hence, we design a novel loss function termed as *Constituency loss* ($\mathcal{L}^{cons}$) to train our model which we describe next.

Constituency loss: This loss ensures that the output of the Segregation Network, $\{u[r], \quad r = 1, ..., K\}$ gives positive values for only those classes which has been mixed to create x_k. Thus, the network is expected to output not only the correct proportion over the *mixing* class set m, but also give zero output over the *non-mixing* class set nm. Based on this requirement, the loss function can be written for the m and nm classes as follows

$$\mathcal{L}^{cons} = \sum_{r \in nm} u[r]^2 + g * \sum_{r \in m} (u[r] - \beta[r])^2 \qquad (1)$$

where, β denotes the mixing coefficient vector which has zeros for the non-mixing classes, and values of α and $(1-\alpha)$ in their relevant places for the mixing classes. Let us define $S^0 = \{\beta[r] \in \mathbb{R} \mid \beta[r] = 0\}$ and $S^{\neq 0} = \{\beta[r] \in \mathbb{R} \mid \beta[r] \neq 0\}$ denote the zero element and non-zero element sets. The set S^0 denotes the sparse set of $\beta[r]$ output for the *non-mixing* classes, while the $S^{\neq 0}$ is for the classes used to form x_k. The weight $g > 1$ plays a significant role in training the model as shown in the ablation studies. This factor is important since $|S^0| >> |S^{\neq 0}|$, where $|.|$ denotes the cardinality of a set. Hence during training, we penalize the errors in wrongly predicting the value of $\{\alpha, 1-\alpha\}$ much more severely as compared to the incorrect prediction of the zero elements. In our implementation, we found the best value of g to be between $1000 - 2000$ in all our experiments.

4.2. Testing Scenario

As mentioned earlier, we assume that a base pre-trained classifier has been fine-tuned on the training classes with a softmax output layer. Here, it has been taken as the AlexNet [22] or VGG16 [39], from where the features are extracted. In the general testing scenario, the test query can come from one of the seen classes or from a novel class. Given a test query, we consider the *Top-N* classes which get the highest output scores from the classifier, i.e. the possibility of the query belonging to one of these classes is high. The goal of the Segregation Network is to consider each of these top classes, and verify whether the query actually belongs to one of those classes. Taking the top few classes is intuitive since (1) if the query belongs to a seen class, its score is usually high and (2) it reduces the computation required for novelty detection using the proposed Segregation Network. If the query is from a novel class, all retrieved classes are obviously wrong. Here we use training class centers, μ_r where $r \in \{1, 2, 3, .., N\}$, where $(N < K)$ as the prototype exemplars. For each query, q, a set of interpolated points is

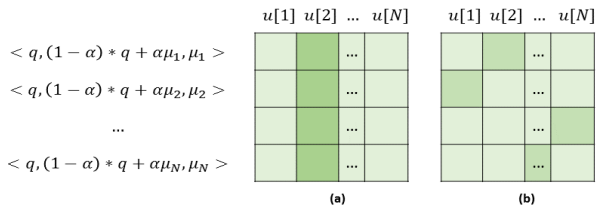

Figure 3. The procedure to compute the membership score for a given query q is shown here. The set of triplet points $\{q, \mu_r, x_r\} \forall r$ is fed to the network and the prediction is represented as a square matrix. (a) and (b) shows two possible cases when the query q comes from the known and novel set respectively. The deeper shade signifies a higher value than the lighter color in the matrix.

generated as $\{q, \mu_r, x_r\}$, where $x_r = (1-\alpha)q + \alpha\mu_r$, which is then is passed through the proposed network. The prediction using all exemplar points $\{\mu_r\}_{r=1}^N$ can be viewed as a square matrix (shown in Figure 3) whose each row signifies the prediction values when paired with a particular cluster center.

The mixing coefficient for the prototype exemplars is kept low while feeding to the model. In other words, the mixing coefficient is kept high for the incoming test data. This is because of the following reasons

- If the query data belongs to one of the known classes, the high α value for the query example would produce a high $u[r]$ value for the correct known class as shown in Figure 3 (a).

- If the query data belongs to an unknown class, the low $(1-\alpha)$ value used for the prototype exemplars forces the network output $u[r]$ to be low for all the (known) classes as shown in Figure 3 (b).

We define the **membership score** as the average of the maximum score in each row. Membership score can be thought of as the likelihood of belonging to the set of known classes. Thus, for a query coming from the known classes, the membership score should be higher as compared to the membership score of a query coming from the novel class set. As seen in Figure 3(a), the individual maximum values of the prediction matrix per row is higher (marked with a deep green shade) and so the membership score (the average value) is higher. When the query belongs to the novel set, as seen in Figure 3(b), the membership score would come out to be lower.

5. Experiments

In this section, we evaluate the proposed *Segregation Network (SN)* and compare against several state-of-the-art approaches. We describe in details the datasets considered for our experiments and the testing protocol. This is fol-

lowed by ablation studies to better understand the proposed approach.

5.1. Datasets Used and Testing Protocol

Here, we report results on two benchmark datasets, namely Caltech 256 [16] and Stanford Dogs [19].

Caltech 256 dataset [16]: This dataset is a standard dataset for visual recognition consisting of objects spread over 256 diverse categories. It consists of $30,607$ images with a minimum of 81 to a maximum of 827 image examples per class. As per the protocol in [32], we took the first 128 classes as known and rest as unknown.

Stanford Dogs dataset [19]: This is a fine grained dataset which has been curated from the ImageNet dataset and consists of images of different breeds of dogs of 120 categories. It consists of $20,580$ images. We consider the first 60 classes, sorted alphabetically as known. The final testing was performed on the remaining 60 classes, similar to the protocol followed in [32].

State-of-the-art Approaches in Literature: We justify the effectiveness of our proposed model for the task of multi-class novelty detection by comparing against the following state-of-the-art approaches: (1) **Finetune [39]:** The fine-tuned network output is taken and thresholded to determine whether a query belongs to the known or novel class; (2) **One-class SVM [37]:** All known classes are considered during training the SVM. During testing the maximum SVM score is considered for determining whether a data-point is novel or not. (3) **KNFST [7]:** The deep features are extracted and normalized and the KNFST algorithm is implemented to project all the training samples into the null space, where all examples from a particular class collapse to a single point. Finally, the distance of a query from the training class in the null space is thresholded to determine whether it is novel or not; (4) **Local KNFST [6]**: This method is similar to that in [7], but focuses only on the training images most similar to the given query for determining its novelty score; (5) **Openmax [3]:** The feature embedding of the penultimate layer of a trained network is taken and mean activation vectors are determined to fit in the Weibull distribution to finally generate a probability vector of dimension $k+1$, where k is the number of classes. (6) **K-extremes [38]:** VGG16 features are extracted and the top 0.1 activation index is used to get the extreme value signatures; (7) **Finetune ($c+C$) [32]:** The network is trained on additional classes from a reference dataset which has examples other than those present in the main dataset; and (8) **Deep Transfer Novelty (DTN):** The state-of-the-art algorithm proposed in [32], where an external dataset as reference data is used to learn negative filters which will not get activated for any of the data from the known categories. The proposed model in [32] is trained by using a novel membership loss function.

Evaluation Protocol: We consider area under the receiver operating characteristic curve (AUC) [10] as the evaluation criteria. AUC is the standard metric used for evaluating these approaches as done in [32].

5.2. Experimental Results

We evaluate our algorithm by using the features extracted from the pre-trained AlexNet [22] and VGG16 [39] networks as done in [32]. Our results are compared against the current state-of-the-art baseline methods and also evaluated on the same set of features. The results for the other approaches are directly taken from [32]. The results as reported in Table 1 show that the proposed model (SN) gives the best result in three out of four cases. Our method with VGG16 features has convincingly outperformed the method in [32] with a margin of 7.9% for Stanford Dogs [19] and 1.3% for Caltech 256 [16] datasets. For AlexNet features, it beats all the other approaches for Stanford Dogs and compares favorably for Caltech 256 dataset. One important point to note is that the proposed framework does not require any external or auxiliary datasets as used in [32] to perform novelty detection. This makes our approach more suitable for real-world applications. Since it does not require any auxiliary dataset, it is also computationally lighter than [32] as we do not need to train any extra network module. We also display the novelty detection results on few test example images from the Caltech 256 dataset in Figure 4. We show successful (a,d) and failure (b,c) cases in Figure 4. Successful cases are the ones where a query known (or unknown) sample is classified to belong to the known (or novel) set.

Table 1. Comparison of the proposed framework (SN) to the state-of-the-art methods using the AUC of the ROC curve evaluation metric. Our algorithm gives convincing results compared to the state-of-the-art Deep Transfer Novelty (DTN), without the use of any extra reference dataset.

Dataset	**Stanford dogs**		**Caltech 256**	
	VGG16	AlexNet	VGG16	AlexNet
FineTune[39]	0.766	0.702	0.827	0.785
One-Class SVM [37]	0.542	0.520	0.576	0.561
KNFST pre [7]	0.649	0.619	0.727	0.672
KNFST [25] [7]	0.633	0.602	0.743	0.688
Local KNFST pre [6]	0.652	0.589	0.657	0.600
Local KNFST [6]	0.626	0.600	0.712	0.628
K-extremes [6]	0.610	0.592	0.546	0.521
OpenMax [3]	0.776	0.711	0.831	0.787
Finetune(c+C) [32]	0.780	0.692	0.848	0.788
DTN [32]	0.825	0.748	0.869	**0.807**
Proposed SN	**0.904**	**0.773**	**0.882**	0.751

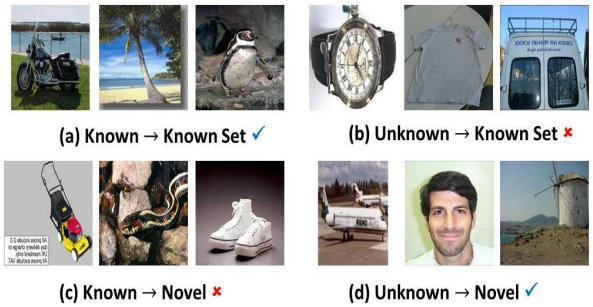

Figure 4. We display some images from the testing set of the Caltech 256 dataset whose novelty has been determined by our SN algorithm. We show cases where our algorithm has been found to be successful (a,d) and also some unsuccessful cases (b,c). The four cases can be described as (a) known example classified as known, (b) unknown classified as novel, (c) known classified as novel and (d) unknown classified to novel set.

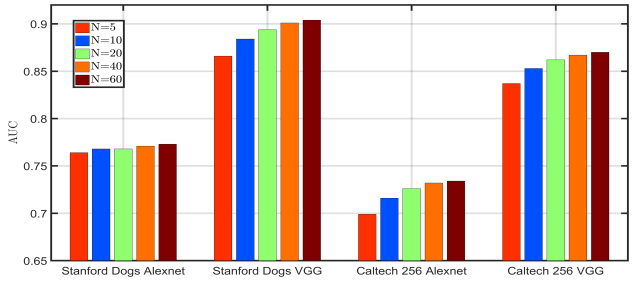

Figure 5. Performance (AUC) of the proposed SN when the number of class prototypes of known classes (N) which is compared with the query is changed. We observe that there is a graceful decrease in AUC when the value of N is slowly decreased for both the datasets using the different feature representations. This naturally leads to a more efficient version of our algorithm.

5.3. Analysis and Observation

We perform extensive analysis to better understand the proposed framework and highlight the salient points.

Effect of number of top classes used for novelty computation: During testing, we compare the query element with all individual class mean prototypes of the known set for determining whether it is novel or not. The results are reported in Table 1 when the prototypes of all the known classes (K) are used during testing. Here, we investigate the effect of taking the prototypes of only the top-N classes (as given by the softmax values from the base network, namely AlexNet and VGG16) and provide the results in Figure 5.

We observe that there is a very gradual monotonic decrease in the performance of our model as the value of N is decreased from 60 to 10. This helps us to develop a computationally lighter version of our algorithm where we get

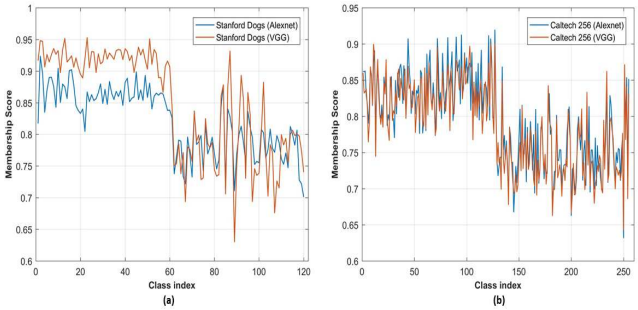

Figure 6. Average class-wise membership scores for the (a) Stanford Dogs and (b) Caltech 256 dataset using the features of the AlexNet [22] and VGG16 [39] network. The first 60 & 128 categories for the two datasets in (a) & (b) are considered in the known class set while the remaining are considered to form the novel classes. We observed that the class-wise membership score for the known class set in general are higher than that of the novel class examples for both the datasets. This clearly demonstrates the effectiveness of our proposed method.

satisfactory performance even when comparing the query with only the top few class prototypes. This becomes especially useful when the number of known classes becomes very large.

Analysis of class-wise membership score: To better understand why our model is working, we plot the average class-wise membership score for the images of the seen and unseen categories for the (a) Stanford Dogs (first 60 categories as known) and (b) Caltech 256 dataset (first 128 categories as known) in Figure 6. Two important conclusions can be drawn from here - (1) the difference in membership score between the seen and novel sets is more pronounced in case of the Stanford Dogs dataset than the Caltech 256 dataset. This effectively means that our algorithm should perform better for the Stanford Dogs dataset which is further reflected in Table 1. (2) The better performance of the VGG16 model over its AlexNet counterpart (as shown in Table 1) can be explained due to the fact that the membership profile for VGG16 has higher peaks and lower troughs than AlexNet. A higher peak and lower trough leads to a better margin for error which results in more ease of determining whether a class is novel or not.

Varying the number of prototypes per class: We have used the mean vector as the class prototype for each class in all our experiments. We conduct experiments on the Stanford Dogs dataset to analyze here how the proposed framework performs when each known class is instead represented by a set of multiple feature vectors instead of a single mean vector. The set of representative feature vectors for each class can be determined by applying k-means algorithm on each category and then selecting the subsequent cluster centers which have been formed. Performance in Table 2 show that there is a very slight increase in perfor-

Table 2. AUC performance of the proposed method SN with increasing number of prototypes taken for each known class during testing on the Stanford Dogs dataset.

	No of prototypes per class			
	1	**2**	**3**	**5**
VGG16	0.904	0.913	0.913	0.914
AlexNet	0.773	0.773	0.775	0.782

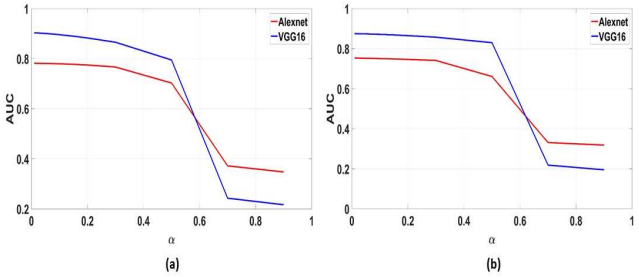

Figure 7. The effect of α on the performance (AUC) for (a) Stanford Dogs and (b) Caltech 256 dataset. Notice that as the value of α is increased the AUC performance drops significantly while the performance saturates at lower values of α.

mance with the increase in the number of prototypes. We thus select a single prototype to evaluate our model during the testing phase.

Effect of mixing coefficient α: Our model is trained by randomly sampling the value α ($0 \leq \alpha \leq 1$) and then assigning it to the pair of points. The purpose of the network is to correctly predict the mixing coefficient. During testing, we set α to a fixed predefined value. In Figure 7, we report the AUC performance of our model for different values of α and observe that the α value of the cluster center μ should be kept low to get better performance. The reason for this being that, the interpolated point x_k, formed by $\alpha\mu + (1-\alpha)q$ with, q being the incoming query, and μ being the class prototype, is much closer to q for a small value of α. Thus if q and μ belong to the same class then the interpolated point should also belong to the same class leading to the trained model outputting a higher prediction value for that particular class. Conversely, if q belongs to a novel class altogether, the interpolated point should be far apart from μ leading the network to give a small value corresponding to that class.

Softmax vs Membership Scores: We analyze the Softmax score along with its membership value for few test cases in Figure 8 for the Stanford Dogs and Caltech 256 dataset respectively. From the figure, it is clear that while the softmax values are overly confident signifying that the image possibly belongs to the known set, our membership score being low helps to determine that it is a novel sample.

Effect of loss weighting factor g: One of the critical design parameters of our algorithm is the choice of the hyperparameter g in equation (1). We observe from Figure 9 the training value loss profile for different values of g for Stanford Dogs dataset when using the VGG16 features. From the figure, we can draw the conclusion that having a weight of 1 i.e., giving equal importance to the zero and non-zero loss gives poor result as the network only learns to output values close to zero. A higher weight assigned to the non-zero element can help to effectively train our model as shown in Figure 9, thereby reducing both the losses.

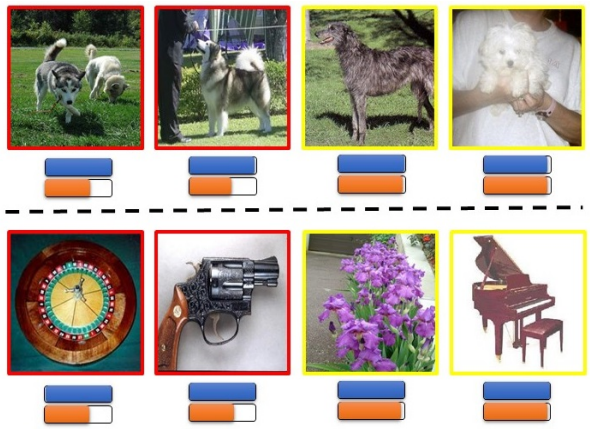

Figure 8. We report the softmax (in blue) and membership score (in orange) for some example test cases from the Stanford Dogs (top row) and Caltech 256 (bottom row). The red (or yellow) bordered images come from the novel (or known) class set. Notice that for a novel query, while softmax values are overly confident, our membership score is low signifying that it is a novel sample.

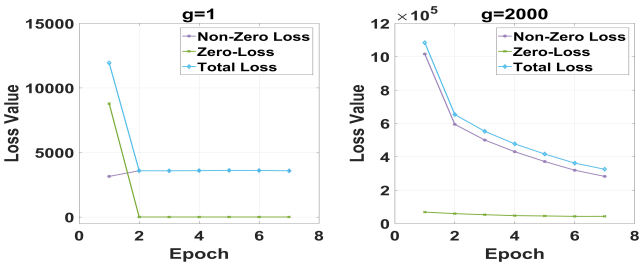

Figure 9. The plot of the zero-element, non-zero element and total loss with epoch for different values of g in equation (1) on Stanford Dogs Dataset.

6. Conclusions

In this paper, we have developed a novel framework for multi-class novelty detection which works using the concept of mix-up technique. We have designed a novel constituency loss for training the proposed model. Experimental evaluations have shown that the model performs favorably with the current state-of-the-art, even without having access to any extra auxiliary dataset. We have also developed an efficient version of our algorithm, without any significant drop in performance, which utilizes the softmax confidence outputs of the pre-trained network.

References

[1] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara. Latent space autoregression for novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 481–490, 2019.

[2] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision(ACCV)*, pages 622–637, 2018.

[3] A. Bendale and T. E. Boult. Towards open set deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2016.

[4] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In *International conference on machine learning (ICML)*, pages 552–560, 2013.

[5] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference Learning Representations (ICLR)*, 2019.

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

[7] 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)*, pages 3374–3381, 2013.

[8] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2015.

[9] 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.

[10] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *International Conference on Machine learning(ICML)*, pages 233–240, 2006.

[11] T. DeVries and G. W. Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[12] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. In *International Conference Learning Representations (ICLR)*, 2017.

[13] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *International Conference on Machine Learning(ICML)*, pages 255–262, 2000.

[14] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[15] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi. Generative openmax for multi-class open set classification. In *British Machine Vision Conference (BMVC)*, 2017.

[16] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[17] D. Ha and D. Eck. A neural representation of sketch drawings. In *International Conference Learning Representations (ICLR)*, 2018.

[18] V. Hautamaki, I. Karkkainen, and P. Franti. Outlier detection using k-nearest neighbour graph. In *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 430–433, 2004.

[19] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2, 2011.

[20] J. Kim and C. D. Scott. Robust kernel density estimation. *Journal of Machine Learning Research*, 13(Sep):2529–2565, 2012.

[21] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The International Journal on Very Large Data Bases*, 8(3-4):237–253, 2000.

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

[23] K. Lee, K. Lee, K. Min, Y. Zhang, J. Shin, and H. Lee. Hierarchical novelty detection for visual object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1034–1042, 2018.

[24] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[25] J. Liu, Z. Lian, Y. Wang, and J. Xiao. Incremental kernel null space discriminant analysis for novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 792–800, 2017.

[26] M. Markou and S. Singh. Novelty detection: a reviewpart 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[27] M. Masana, I. Ruiz, J. Serrat, J. van de Weijer, and A. M. Lopez. Metric learning for novelty and anomaly detection. In *British Machine Vision Conference (BMVC)*, 2018.

[28] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems(NIPS)*, pages 5040–5048, 2016.

[29] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *International Conference on Machine Learning(ICML)*, pages 2391–2400, 2017.

[30] R. Mohammadi-Ghazi, Y. M. Marzouk, and O. Büyüköztürk. Conditional classifiers and boosted conditional gaussian mixture model for novelty detection. *Pattern Recognition*, 81:601–614, 2018.

[31] P. Perera, R. Nallapati, and B. Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2898–2906, 2019.

[32] P. Perera and V. M. Patel. Deep transfer learning for multiple class novelty detection. In *IEEE Conference on Computer*

Vision and Pattern Recognition(CVPR), pages 11544–11552, 2019.

[33] S. Pidhorskyi, R. Almohsen, and G. Doretto. Generative probabilistic novelty detection with adversarial autoencoders. In *Advances in Neural Information Processing Systems(NIPS)*, pages 6822–6833, 2018.

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

[35] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. Adversarially learned one-class classifier for novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3379–3388, 2018.

[36] M. S. Sadooghi and S. E. Khadem. Improving one class support vector machine novelty detection scheme using nonlinear features. *Pattern Recognition*, 83:14–33, 2018.

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

[38] A. Schultheiss, C. Käding, A. Freytag, and J. Denzler. Finding the unknown: Novelty detection with extreme value signatures of deep neural activations. In *German Conference on Pattern Recognition*, pages 226–238. Springer, 2017.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] J. P. Stevens. Outliers and influential data points in regression analysis. *Psychological Bulletin*, 95(2):334, 1984.

[41] Y. Tokozume, Y. Ushiku, and T. Harada. Learning from between-class examples for deep sound recognition. In *International Conference Learning Representations (ICLR)*, 2017.

[42] Y. Tokozume, Y. Ushiku, and T. Harada. Between-class learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5486–5494, 2018.

[43] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[44] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. Online unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.

[45] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference Learning Representations (ICLR)*, 2018.

[46] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference Learning Representations (ICLR)*, 2018.