Multi-Class Novelty Detection with Generated Hard Novel Features

Wei-Ta Chu
wtchu@gs.ncku.edu.tw
Wei-Ting Cao
a0903511820@gmail.com

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

Abstract

It has been shown that convolutional neural networks clearly have the overconfidence problem, i.e., mis-classify a novel sample into one of the known classes with high confidence. The task of multi-class novelty detection is thus important to detect novel samples during inference. In this work, we propose to generate hard novel features via a generative adversarial network to facilitate constructing a powerful novelty detector. The generated features should be around the boundaries between known classes and novel classes. They cause a bigger challenge for the novelty detector, and consequently enforce the novelty detector to be stronger. We verify effectiveness of hard novel features from several perspectives, and show that this idea yields the state-of-the-art performance.

1 Introduction

Deep neural networks (DNNs) have been widely developed for image recognition and image classification. Most networks work under a close-set assumption, and classify every query sample into one of the training classes. A novel sample coming from a completely different category might be classified into one of these known classes with high confidence [11][6]. However, a DNN practical to real applications should be able to identify whether a given query sample is from the trained classes, and then try to classify it.

Novelty detection targets at determining whether a query sample belongs to the known classes. This task is generally more challenging than out-of-distribution detection because novel samples may come from a distribution similar to known classes [16]. When the number of known classes is more than one, it is called a *multi-class novelty detection* (MCND) problem [16]. In this paper, we focus on the MCND problem.

Motivated by hard negative samples generation in image classification [18] and image recognition [9], we propose to generate *hard novel features* to build a more robust novelty detector. Given training samples from known classes, we extract features by the penultimate layer of convolutional neural networks (CNNs) like AlexNet [10] and VGG16 [17]. A generative adversarial network (GAN) is trained to generate features from noise vectors such that the distribution of generated features is similar to the features extracted from known classes, named known features in the following. The main point is that the generated features are not only novel to known classes, but also distribute around the boundary between known classes

and novel classes. Given known features and generated features, a classifier is constructed to be a novelty detector.

Overall, contributions of this work is twofold. 1) We develop a generative adversarial network to generate hard novel features so that a more robust novelty classifier can be constructed. 2) The generated hard novel features can also be utilized to boost other multi-class novelty detection methods. We verify performance gain from several perspectives.

2 Related Works

Several problems have similar habitude but with subtle difference, including out-of-distribution (OOD) detection [4], novelty detection (ND) [1], and open-set recognition (OSR) [2]. OOD detection determines whether a query comes from an in-class distribution (a specific dataset) or other unseen ones (other datasets). Novelty detection targets at identifying whether a given query sample is from a novel class, which is unseen to the classifier when it was trained. Novel samples and known samples may be from the same or similar distributions. Multi-class novelty detection (MCND) is a branch of novelty detection, while in MCND there are multiple known classes. OSR is similar to novelty detection, where the probability of an input being from an unknown class is to be estimated. In addition to keeping track of the errors between known and unknown categories, OSR also needs to take care of the errors of the standard multi-class classification over known categories.

We briefly review the most recent works about MCND in the following. Based on deep neural networks, the activation patterns of the penultimate layer of the network is considered, and an OpenMax layer was proposed to estimate the probability of an input being from an unknown class [2]. This method was originally designed for the OSR task, and can be modified to conduct MCND. Perera and Patel [16] proposed a transfer learning-based method to consider knowledge of an out-of-distribution dataset (reference dataset) to improve MCND. Globally negative filters were learnt in the end-to-end approach to discriminate novel samples. To avoid the requirement of the reference dataset, most recently the same research group proposed to utilize class-specific activation patterns of image patches, and jointly considered local and global information to train the novelty detection network [14]. Concurrently, they further studied MCND especially considering the problem of data distribution shift [15], by combining techniques of novelty detection and domain adaptation. Also without the need of the reference dataset, Bhattacharjee et al. [3] used the mix-up technique for novelty detection. A segregation network was developed to estimate the proportions how an input sample is mixed with known classes. In this work, we are in line with MCND without data distribution shift. We propose to generate hard novel features to improve novelty detection, and will compare with the state of the arts [16][14][3].

3 Generating Hard Novel Features

3.1 Overview

Let $X \in \mathbb{R}^{d \times N}$ denote the input data, where d is the dimensionality of a feature vector, and N is the number of training samples. The label of each data sample is a K-dimensional one-hot vector (K known classes). The set of known classes is denoted as $C = \{C_1, C_2, ..., C_K\}$. During the testing stage, given a query, the objective is to determine whether it belongs to one of the known classes or a novel class.

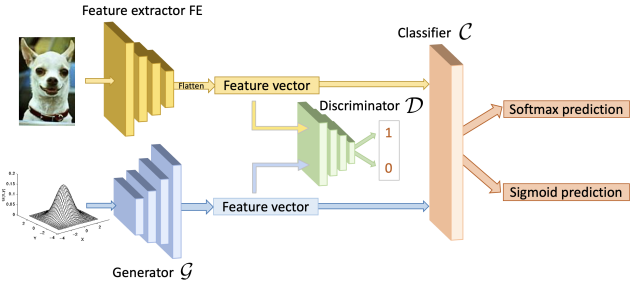


Figure 1: Illustration of the proposed network (named as HNF+Classifier).

Fig. 1 illustrates the proposed network. Given the set of training samples $X = \{X_1, ..., X_N\}$ in known classes, the top branch consisting of backbone convolutional neural networks to extract *known features* $F = \{f_1, ..., f_N\}$. In the bottom branch, a generator $\mathcal{G}$ is constructed to map latent variables z's to outputs $G = \{\mathcal{G}(z_i)\}$, which recover the distribution of F as much as possible. A discriminator $\mathcal{D}$ is constructed to output the probability that a sample is from the in-class distribution. Both known features and generated features are fed to a novelty classifier $\mathcal{C}$, which outputs K confidence values corresponding to K known classes. If the confidence value of the maximum class is less than a threshold, the input feature is viewed as novel; otherwise, the input feature is classified into one of the known classes. The confidence scores of generated features G are expected to be lower than the threshold. Conceptually, the generated features distribute similarly to but outside known features. For the classifier $\mathcal{C}$, the generated features are harder to be classified. These features are thus called "hard novel features" (HNF in short) in this paper. To train this network, we design loss functions in the following.

3.2 Adversarial Loss

The generator $\mathcal{G}$ is trained to generate features that are around the boundary of the distribution formed by the known features $x_i \in X$. Let $P_k(x)$ denote the distribution of known features. Given noise vectors z from a prior distribution $P_{pri}(z)$, the generator $\mathcal{G}$ generates features $\mathcal{G}(z)$. The discriminator $\mathcal{D}$ is trained to distinguish whether an input vector is from the distribution of known classes or from the generated feature distribution. To train an adversarial network, the following min-max objective is to be optimized:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathrm{E}_{P_k(x)}[\log \mathcal{D}(x)] + \mathrm{E}_{P_{pri}(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]. \tag{1}$$

This generator tends to generate the feature distribution similar to the known distribution $P_k(x)$. However, our goal is to generate novel features similar to but not inside the known distribution. To achieve this, results of the classifier $\mathcal{C}$ on the generated features are forced to be similar to a uniform distribution $\mathcal{U}$. Inspired by [12], we set the adversarial loss as:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \underbrace{\mathrm{E}_{P_k(x)}[\log \mathcal{D}(x)] + \mathrm{E}_{P_{pri}(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]}_{\text{(a)}} + \underbrace{\mathrm{E}_{P_{pri}(z)}[KL(\mathcal{U} || P_{\mathcal{C}}(\mathcal{G}(z)))]}_{\text{(b)}}, \tag{2}$$

where $P_{\mathcal{C}}(\mathcal{G}(z))$ denotes the distribution of classification results on generated features $\mathcal{G}(z)$, and KL denotes the Kullback-Leibler divergence. The term (a) is the original adversarial

loss mentioned in eqn. (1). The term (b) forces the generator to generate features so that the predictive distribution on generated features is closer to the uniform one.

3.3 Confidence Loss

To construct a classifier, the given input features are usually processed by fully-connected layers, and the last layer is followed by a softmax layer to output the probability of the input being each class. Typically the loss function is the cross entropy between prediction and the ground truth so that known features can be classified into one of known classes. The classifier shown in Fig. 1 acts as a novelty detector, and we hope the classification results should be similar to the uniform distribution (zero confidence) if the input is a generated HNF. Therefore, the loss function for the classifier $\mathcal{C}$ is defined as:

$$\min_{\theta} \underbrace{\mathrm{E}_{P_{pri}(z)}[KL(\mathcal{U}||P_{\mathcal{C}}(\mathcal{G}(z)))]}_{\text{(b)}} + \underbrace{\mathrm{E}_{P_k(x)}[-\log P_{\mathcal{C}}(x)]}_{\text{(c)}}, \tag{3}$$

where the term (b) is the same as the term (b) in eqn. (2), and the term (c) is the conventional cross entropy loss calculated based only on features x extracted from known classes. The term θ is the network parameters of the classifier.

Although softmax is usually used in a classifier, the work [16] pointed out its limitations when it is adopted in a threshold-based novelty classifier. First, a model trained using cross-entropy loss may produce low activation values for known classes during inference. This causes a problem because samples of known classes may be mis-classified as false negatives. Second, the softmax normalization is done across all classes. Based on softmax activations and the cross-entropy loss, inaccurate cross-class relationships may be encouraged.

To tackle with this problem, the last part of the classifier is branched into two parts. The first part keeps adopting softmax as the activation function, and outputs the probabilities of the given input belonging to each class. The second part adopts sigmoid as the activation function, and we can view the output values as the confidence of the input image belonging to each individual class. We denote outputs of the softmax-based classifier and the sigmoid-based classifier as $P_{\mathcal{C}_e}(x)$ and $S_{\mathcal{C}_\sigma}(x)$, respectively. Overall, the loss function is re-written as:

$$\min_{\theta} \underbrace{\mathrm{E}_{P_{pri}(z)}[KL(\mathcal{U}||P_{\mathcal{C}_e}(\mathcal{G}(z)))]}_{\text{(b)}} + \alpha \underbrace{\mathrm{E}_{P_{pri}(z)}[\|S_{\mathcal{C}_\sigma}\|_2]}_{\text{(b')}} + \underbrace{\mathrm{E}_{P_k(x)}[-\log P_{\mathcal{C}_e}(x)]}_{\text{(c)}} + \alpha \underbrace{\mathrm{E}_{P_k(x)}[-\log S_{\mathcal{C}_\sigma}]}_{\text{(c')}}, \tag{4}$$

where $\|S_{\mathcal{C}_\sigma}\|_2$ denotes the L2 norm of $S_{\mathcal{C}_\sigma}$. The parameter α is set to control the weighting of the sigmoid-based classifier. The term (b)' makes the sigmoid values of an HNF to all classes approach zeros as much as possible.

3.4 Overall Objective

An alternating training scheme [12] is utilized to train the network illustrated in Fig. 1. In each training epoch, parameters of the generator $\mathcal{G}$ and discriminator $\mathcal{D}$ are first adjusted based on a pre-trained feature extractor and a default classifier $\mathcal{C}$. After that, parameters of the feature extractor and the classifier are adjusted. Conceptually these two parts improve

each other. Overall, the joint objective function is defined as follows.

$$\min_{\mathcal{G}}\max_{\mathcal{D}}\min_{\theta} \underbrace{\mathrm{E}_{P_k(x)}[\log \mathcal{D}(x)] + \mathrm{E}_{P_{pri}(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]}_{\text{(a)}} + \underbrace{\mathrm{E}_{P_{pri}(z)}[KL(\mathcal{U}||P_{\mathcal{C}_e}(\mathcal{G}(z)))]}_{\text{(b)}} + \underbrace{\alpha \mathrm{E}_{P_{pri}(z)}[\|S_{\mathcal{C}_\sigma}\|_2]}_{\text{(b')}} + \underbrace{\mathrm{E}_{P_k(x)}[-\log P_{\mathcal{C}_e}(x)]}_{\text{(c)}} + \underbrace{\alpha \mathrm{E}_{P_k(x)}[-\log S_{\mathcal{C}_\sigma}]}_{\text{(c')}}. \tag{5}$$

When testing, features q of a test sample are extracted by the feature extractor, and then are processed and input to the sigmoid-based classifier to output activation scores $P_{\mathcal{C}_\sigma}(q)$ corresponding to each class. The class with the maximum score which is larger than the threshold is viewed as the predicted class.

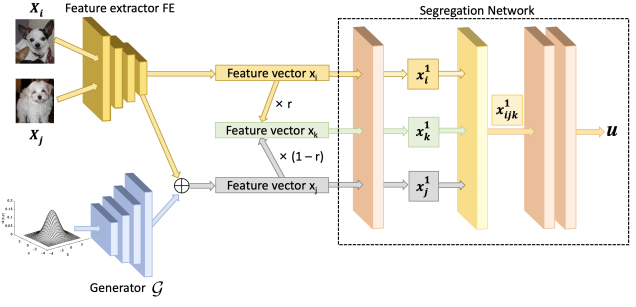


Figure 2: Illustration of integrating hard novel features into the segregation network.

4 Incorporation with the Segregation Network

4.1 Segregation Network

The classifier $\mathcal{C}$ in Fig. 1 is already a multi-class novelty detector. Here we further emphasize that the hard novel features can be seamlessly incorporated with the state-of-the-arts methods to provide more performance gain. The most recent methods include UPCAP [14] and the segregation network approach [3]. Compared to [14], programs of the segregation network approach are publicly available, and we especially take it as the main instance in the following.

The segregation network consists of three fully connected (fc) layers with ReLU activations and dropout between each layer, except for the final fc layer. The final layer with the sigmoid activation function outputs a K-dimensional vector representing the estimated proportions of how the input features are mixed by features coming from known classes.

Figure 2 illustrates the segregation network. The segregation network takes a triplet set of feature vectors $\{x_i, x_j, x_k\}$ as inputs, where x_i and x_j are features extracted from known classes and x_k is the mixture interpolated by x_i and x_j with an arbitrary proportion. Namely, $x_k = rx_i + (1-r)x_j, r \in (0,1)$. Let's define the outputs of the first fc layer, which is shared by the triplet inputs, as $\{x_i^1, x_j^1, x_k^1\}$. We concatenate them together to form x_{ijk}^1 and feed it forward through the rest of the two fc layers. The final output given by the last fc layer is denoted as $u = [0,1]^K$, where each element denotes the proportion by which the mixed sample x_k has been constructed from that class. The main idea of the segregation network is

that, if both x_i and x_j come from known classes, the proportions for mix-up can be estimated well. But if x_j, for example, comes from novel classes, the proportion estimated for x_j should be close to zero. Assume that there are four known classes. If x_k is mixed by 80% of x_i coming from the first class and 20% of x_j coming from the third class, i.e., $x_k = 0.8x_i + 0.2x_j$, the output values should approximately be $u = (0.8, 0, 0.2, 0)$. On the other hand, if x_k is mixed by x_i coming from the first class and x_j coming from a novel class, the output values should approximately be $u = (0.8, 0, 0, 0)$. When integrating HNF into the segregation network, x_k may be mixed by x_i coming from the first class and x_j coming from one HNF. The output values for x_k should still be $u = (0.8, 0, 0, 0)$.

In [3], a constituency loss was proposed to ensure the output vector u gives positive values for the known classes mixed to create x_k. The set of known classes used to mix is call a mixing set. The network is expected to output not only the correct proportions for the mixing set M, but also give zeros for non-mixing set $\overline{M}$. Specifically, the loss is:

$$L_{cons} = \sum_{m \in \overline{M}} u[m]^2 + g \times \sum_{m \in M} (u[m] - v[m])^2 \tag{6}$$

where v denotes the mixing coefficient vector which has zeros for the classes in $\overline{M}$, and values of r and $(1-r)$ for the classes in M. The parameter g is usually set larger than 1 because $|\overline{M}| \gg |M|$, where $|\cdot|$ denotes the cardinality of a set. We penalize the errors in the second term much more severely as compared with the incorrect prediction of the first term. The value of g is empirically set as 2,000 in our experiments.

4.2 Testing Scenario

Let $\mu_1, \mu_2, ..., \mu_K$ denote the center vectors of K known classes. For a query feature q, we make a linear interpolation of the query and each class center, and get $x_q^{(i)} = r\mu_i + (1-r)q$, where $i \in 1, 2, ..., K$. The set of triplets $\{\mu_i, q, x_q^{(i)}\}$ is fed to the segregation network to predict K mixing vectors. For $x_q^{(i)}$, the mixing vector is predicted as $u_q^{(i)} = (u_1^{(i)}, u_2^{(i)}, ..., u_K^{(i)})$. These values $u_j^{(i)}$'s are the likelihood values of $x_q^{(i)}$ belonging to K known classes.

For the query q, we have K interpolation counterparts $x_q^{(1)}, ..., x_q^{(K)}$, and we can get K mixing vectors $u_q^{(1)}, ..., u_q^{(K)}$. The score of the query q belonging to the first class is measured by

$$s_1 = w^{(1)} \times u_1^{(1)} + w^{(2)} \times u_1^{(2)} + \cdots + w^{(K)} \times u_1^{(K)}. \tag{7}$$

In [3], all parameters $w^{(1)}$ to $w^{(N)}$ are set as $\frac{1}{K}$. Similarly, we can get the query q belonging to the second to the Kth classes as $s_2, s_3, ..., s_K$. Overall, the query q most likely belongs to the class i^*, where $i^* = \arg\max_i s_i$. If the score s_{i^*} is lower than a threshold τ, we say the query q is a novel sample. In Sec. 5 we measure performance by area under ROC curve, where the threshold τ is dynamically changed to incur different precisions and recalls.

5 Experiments

5.1 Datasets

We evaluate the proposed method based on four benchmark datasets. The Stanford Dogs dataset [8] is a fine-grained dataset that includes 20,580 images of 120 different dog breeds.

Sorted alphabetically, the first 60 classes are taken as known classes, and the rest is taken as novel classes. The Caltech 256 dataset [5] includes 30,607 images of 256 diverse categories. As per the protocol [16], we take the first 128 classes as known classes, and the rest as novel classes. The CUB 200 dataset [19] includes 6,033 images belonging to 200 distinct bird categories. We sort names of the bird categories alphabetically and use the first 100 classes as the known classes, and the remaining classes are viewed as novel. The FounderType-200 Dataset [13] is a collection of Chinese character images in different font types. There are 200 different font-types with 6,763 images from each class. The first 100 classes are used as the known classes, and the remaining 100 classes are used as novel. We use area under the receiver operating characteristic curve (AUC) to evaluate the performance, as done in previous works [16][3][14].

5.2 Network Architectures

Network architectures of the generator and the discriminator shown in Figure 1 are given as follows. The generator consists of six blocks. Except for the last block, each block includes a convolutional layer with 1×1 convolutional kernel, a batch normalization layer, and a fully-connected layer with the leakyReLU activation function. The last block is constituted by one convolutional layer with the tanh activation function. A given 100-dimensional noise vector is input to the generator, and the six blocks transform it into dimensions of 128, 256, 512, 1024, 2048, and 4096, respectively. The discriminator is constructed by three fully-connected layers, where the first two layers are with the leakyReLU activation, and the last layer simply outputs the linearly processing results. A given 4096-dimensional vector is transformed by these three layers into dimensions of 512, 512, and 1, respectively.

5.3 Experimental Results

Performance Comparison. Same as previous works [16], we evaluate the proposed method by using the features extracted by AlexNet [10] and VGG-16 [17] pre-trained on the ImageNet dataset. Performance comparison with existing works is shown in Table 1. The "HNF+Classifier" row shows performance of the method described in Sec. 3, and the "HNF+SN" row shows performance of the segregation network aided by our generated HNFs (Sec. 4). The "HNF+SN*" row shows performance of the segregation network aided by our generated HNFs, but the parameters $w^{(1)}$ to $w^{(N)}$ in eqn. (7) are set as the probabilities output by the classifier shown in Fig. 1.

As shown in the "HNF+Classifier" row, the generated HNFs effectively assist a simple threshold-based novelty detector, and outperforms the state of the arts DT [16], SN [3], UPCAP [14], and even UPCAP† [14] (which takes information from a reference dataset). Comparing SN with "HNF+SN", we found that HNFs effectively aid the segregation network to get performance gain. It can be further improved if the mixing coefficients are set as predicted probabilities by the baseline novelty classifier ("HNF+SN*").

Using AlexNet and VGG-16 as the backbones for feature extraction is to fairly compare performance with existing methods. It would be interesting to check if more performance gains can be obtained when we use more advanced backbones. Table 2 shows AUCs obtained by the HNF+Classifier approach on the Stanford Dogs dataset, when different backbones are used as the feature extractors. As can be seen, when more advanced networks like ResNet-101, ResNet-152 [7], or ResNeXt-50 [20] are used for feature extraction, performance of

Table 1: Performance comparison in terms of AUC. Our method consistently improves the state of the arts. Values of compared baselines are from the original papers. The values with † in the SN row are obtained by our implementation.

Methods	Stanford Dogs		Caltech 256		CUB		FounderType		Overall
	VGG-16	AlexNet	VGG-16	AlexNet	VGG-16	AlexNet	VGG-16	AlexNet	
Fine-tuned CNN	0.766	0.702	0.827	0.785	0.931	0.909	0.841	0.650	0.801
Open-Max [1]	0.776	0.711	0.831	0.787	0.935	0.915	0.852	0.667	0.809
Finetune ($c+\bar{C}$)	0.780	0.692	0.848	0.788	0.921	0.899	0.754	0.723	0.801
DT [14]	0.825	0.748	0.869	0.807	0.958	0.947	0.893	0.741	0.849
SN [1]	0.904	0.773	0.828	0.751	0.954†	0.926†	0.940†	0.931†	0.876
UPCAP [14]	0.827	0.751	0.859	0.826	0.972	0.952	0.876	0.798	0.858
UPCAP† [14]	0.873	**0.812**	0.870	0.847	0.979	0.965	0.898	0.801	0.881
HNF+Classifier	0.905	0.798	**0.917**	**0.853**	**0.985**	**0.974**	0.955	**0.946**	**0.916**
HNF+SN	0.912	0.797	0.841	0.762	0.961	0.938	0.956	0.937	0.888
HNF+SN*	**0.917**	0.802	0.846	0.766	0.964	0.942	**0.959**	0.941	0.892

Table 2: AUCs of novelty detection by the HNF+Classifier approach, based on the Stanford Dogs dataset and taking different backbones as the feature extractors.

	AlexNet	VGG-16	ResNet-50	ResNet-101	ResNet-152	ResNeXt-50 (32 x 4d)
AUC	0.798	0.905	0.907	0.921	0.921	0.919

the HNF+Classifier approach can be improved further. This shows the potential and future extension of the proposed hard novel feature generation.

Ablation Studies. Table 3 shows performance variations of "HNF+Classifier" in different settings, based on the Stanford Dogs dataset and taking AlexNet as the feature extractor. Taking Fig. 1 as the reference, the softmax only version (first row) means only the top branch is enabled, and the last layer is softmax. Similarly, the sigmoid only version (second row) means only the top branch is enabled, and the last layer is sigmoid. The fourth row shows that both branches of feature extraction and HNF generation are enabled, and the last layer is softmax. Comparing the first row with the fourth row, we clearly see that the generated novel features boost AUC from 0.702 to 0.758. Comparing the fourth row with the last row, we see that if both softmax-based and sigmoid-based predictions are considered, the AUC is further boosted to 0.793.

Table 4 shows performance variations when the weight α in eqn. (5) to integrate the sigmoid-based terms is varied, also based on four datasets and using AlexNet as the feature extractor. Consistently, when α is set as 20, the best AUC can be obtained. This shows the impact of the sigmoid activation scores, comparing to softmax.

In Sec. 3.3, to train the hard novel feature generator, the classification results are guided to be similar to the uniform distribution. To verify effectiveness of this factor, we compare the proposed full model (eqn. (5)) and the version trained based on eqn. (5) but without the term (b). Table 5 shows performance variations of these two versions, based on the Stanford Dogs dataset and using VGG-16 as the feature extractor. As can be seen, the impact of this

Table 3: Performance variations of "HNF+Classifier" in different settings.

Softmax	Sigmoid	Hard Novel Features	AUC
√	-	-	0.702
-	√	-	0.698
√	√	-	0.733
√	-	√	0.758
-	√	√	0.768
√	√	√	**0.798**

Table 4: Performance variations when the weight α in eqn. (5) to integrate the sigmoid-based terms is varied.

α	1	5	10	20	30
Stanford Dogs	0.765	0.773	0.776	**0.798**	0.791
Caltech	0.826	0.834	0.845	**0.853**	0.853
CUB	0.948	0.967	0.970	**0.974**	0.971
FounderType	0.868	0.911	0.931	**0.946**	0.944

Table 5: Performance variations when the term (b) in eqn. (5) is considered or not.

Methods	AUC
Without term (b) in eqn. (5)	0.860
With term (b) in eqn. (5)	**0.905**

KL divergence term is significant, boosting AUC from 0.860 to 0.905.

The generated HNFs play a role similar to features extracted from the reference dataset in the DT method [16]. However, because the novelty detector can more effectively learn the boundary between known classes and novel classes based on HNFs, we may need much fewer novel features to learn the detector. The Stanford Dogs dataset includes 5,312 images. In the DT method, they used the ImageNet validation datasets (totally 50,000 fully-labeled images) as the reference to train their framework. There are thus 55,312 images in total for training. In our proposed method, we can generate only 5,312 HNFs, and thus totally 10,624 feature vectors are used for training.

In a Windows 10 system equipped with RTX-2070 GPU and AMD 3600x CPU, we implement the DT method and our method by Pytorch in a single thread. The mini-batch size is set as 64. Based on the respective training set, we evaluate the training time per epoch for both methods, and show results in Table 6. As can be seen, because the volume of training set for our method is much smaller, our training time is much less but the obtained AUC is clearly better than the DT method. This result further inspires us to study how the volume of hard novel features influences performance. Table 7 shows performance variations when different numbers of hard novel features are used for training. For Stanford Dogs dataset that consists of 5,312 images, the ratio 50%, for example, means that we use the extracted 5,312 features and $5,312 \times 0.5 = 2,656$ generated HNFs together as the training data. The results show that consistently the best AUCs are obtained when the same number of hard novel features as known features are taken for training. In the whole evaluation section, we use the 1:1 setting to show experimental results.

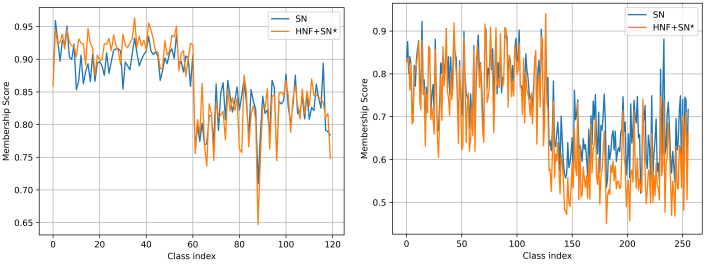
Fig. 3 shows average class-wise membership scores for the Stanford Dogs dataset and the Caltech 256 dataset, using AlexNet as the feature extractor. In Fig. 3(a), the first 60 classes are considered as known classes, while the remaining are novel classes. Overall, the average predicted scores of known classes by the basic SN method [3] and the HNF+SN* method are 0.900 and 0.919 (p-value < 0.0001), respectively. And the average scores of novel classes by two methods are 0.823 and 0.817 (p-value < 0.057), respectively. As can be seen, we see that training SN with HNFs (orange curve) makes scores for known classes higher and makes scores for novel classes lower. The same trend can also be seen in Fig. 3(b), where

Table 6: Comparison of training time per epoch based on the DT method and our method.

	DT	HNF+Classifier
Time per epoch	390 s	52 s
AUC	0.748	0.793

Table 7: Performance variations when different numbers of HNFs are used for training.

Ratios	DT	5%	10%	25%	50%	100%	150%
Stanford Dogs	0.748	0.774	0.778	0.785	0.788	**0.798**	0.786
Caltech	0.807	0.832	0.835	0.845	0.850	**0.853**	0.852
CUB	0.947	0.964	0.967	0.971	0.972	**0.974**	0.973
FounderType	0.741	0.931	0.934	0.939	0.942	**0.946**	0.941



(a) (b)

Figure 3: Average class-wise membership scores for the (a) Stanford Dogs dataset and (b) the Caltech 256 dataset.

the first 128 classes are considered as known classes.

Classification Performance. The constructed novelty detector (HNF+Classifier) can be directly adopted to do image classification. Following the setting of DT [16], under a close-set assumption, we only test samples from known classes. The classifier outputs how likely a given test sample belongs to each class. The class with the highest likelihood value is viewed as the classification result. Based on the Stanford Dogs dataset with the VGG-16 as the backbone, Table 8 shows classification accuracies obtained by the pre-trained VGG-16, fine-tuned VGG-16, DT, and HNF+Classifier. It can be clearly seen that the performance improvement brought by the generated HNFs.

6 Conclusion

Inspired by out-of-distribution data generation, we build a GAN to generate hard novel features as well as a multi-class novelty detector to achieve the state-of-the-art performance. Without the need of a reference dataset, a GAN is constructed to generate features around the boundaries of known classes. These generated features are viewed as novel features and are harder to be classified into known or novel. This characteristics increases the challenge to train the novelty detector, and enforces the constructed detector to be more powerful. The generated hard novel features can also be incorporated with existing methods like the segregation network to provide performance gain. In the evaluation, we present comprehensive performance comparison and ablation studies. Some interesting characteristics like the volume of generated features are also studied.

Table 8: Classification accuracies on the Stanford Dogs dataset based on different methods.

	Pre-trained VGG-16	Fine-tuned VGG-16	DT [16]	HNF+Classifier
Accuracy	0.73	0.76	0.80	0.83

Acknowledgement. This work was funded in part by Qualcomm through a Taiwan University Research Collaboration Project and in part by the Ministry of Science and Technology, Taiwan, under grants 110-2221-E-006-127-MY3, 108-2221-E-006-227-MY3, 107-2923-E-006-009-MY3, and 109-2218-E-002-015.

References

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

[2] A. Bendale and T.E. Boult. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1563–1572, 2016.

[3] S. Bhattacharjee, D. Mandal, and S. Biswas. Multi-class novelty detection using mix-up technique. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision*, pages 1400–1409, 2020.

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

[5] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[6] C. Guo, G. Pleiss, Y. Sun, and K.Q. Weinberger. On calibration of modern neural networks. In *Proceedings of International Conference on Machine Learning*, 2017.

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

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

[9] T. Kim, K. Hong, and H. Byun. The feature generator of hard negative samples for fine-grained image recognition. *Neurocomputing*, 2020.

[10] A. Krizhevsky, I. Sutskever, and G.E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of Advances in Neural Information Processing Systems*, 2017.

[12] K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *Proceedings of International Conference on Learning Representations*, 2018.

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

[14] P. Oza and V.M. Patel. Utilizing patch-level category activation patterns for multiple class novelty detection. In *Proceedings of European Conference on Computer Vision*, pages 421–437. Springer, 2020.

[15] P. Oza, H.V. Nguyen, and V.M. Patel. Multiple class novelty detection under data distribution shift. In *Proceedings of European Conference on Computer Vision*, pages 432–449. Springer, 2020.

[16] P. Perera and V.M. Patel. Deep transfer learning for multiple class novelty detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 11544–11552, 2019.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations*, 2015.

[18] S. Vandenhende, B. De Brabandere, D. Neven, and L. Van Gool. A three-player gan: Generating hard samples to improve classification networks. In *Proceedings of International Conference on Machine Vision Applications*, 2019.

[19] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[20] S. Xie, R. Girshick, P. Dollar, Tu Z., and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.