electronics

MDPI

Article

Improving the Performance of Open-Set Recognition with Generated Fake Data

András Pál Halász *, Nawar Al Hemeary, Lóránt Szabolcs Daubner, Tamás Zsedrovits and Kálmán Tornai

Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, 1083 Budapest, Hungary
* Correspondence: halasz.andras@itk.ppke.hu

Abstract: Open-set recognition models, in addition to generalizing to unseen instances of known categories, have to identify samples of unknown classes during the training phase. The main reason the latter is much more complicated is that there is very little or no information about the properties of these unknown classes. There are methodologies available to handle the unknowns. One possible method is to construct models for them by using generated inputs labeled as unknown. Generative adversarial networks are frequently deployed to generate synthetic samples representing unknown classes to create better models for known classes. In this paper, we introduce a novel approach to improve the accuracy of recognition methods while reducing the time complexity. Instead of generating synthetic input data to train neural networks, feature vectors are generated using the output of a hidden layer. This approach results in a less complex structure for the neural network representation of the classes. A distance-based classifier implemented by a convolutional neural network is used in our implementation. Our solution's open-set detection performance reaches an AUC value of 0.839 on the CIFAR-10 dataset, while the closed-set accuracy is 91.4%, the highest among the open-set recognition methods. The generator and discriminator networks are much smaller when generating synthetic inner features. There is no need to run these samples through the first part of the classifier with the convolutional layers. Hence, this solution not only gives better performance than generating samples in the input space but also makes it less expensive in terms of computational complexity.

Keywords: open-set recognition; sample generation; distance-based classification; GAN

Citation: Halász, A.P.; Al Hemeary, N.; Daubner, L.S.; Zsedrovits, T.; Tornai, K. Improving the Performance of Open-Set Recognition with Generated Fake Data. *Electronics* **2023**, *12*, 1311. https://doi.org/10.3390/electronics12061311

Academic Editors: Juan M. Corchado, Byung-Gyu Kim, Carlos A. Iglesias, In Lee, Fuji Ren and Rashid Mehmood

Received: 2 January 2023
Revised: 25 February 2023
Accepted: 7 March 2023
Published: 9 March 2023

1. Introduction

Nowadays, various machine learning methods provide excellent results in different classification and object recognition tasks. These methods reach or even exceed the performance of humans in numerous cases. One of the examples is the best-performing model on the Modified National Institute of Standards and Technology (MNIST) dataset [1], achieving an error rate of 0.21% [2]. Based on these impressive achievements, this field no longer contains challenges. However, the experiments yielding these results were conducted in closed-set scenarios, assuming that all possibly occurring classes are known during the training phase of the models. The open-set case is a much more realistic scenario, where new classes can appear after optimizing the model's parameters. Hence, during the evaluation of the method, the data samples belonging to previously unknown classes must be rejected. Traditional recognition algorithms fail to solve the task even if the dataset is precisely classified in a closed-set setting.

In this study, we abandoned the idea of partitioning the space of data (or features) with hyperplanes because most of the possible inputs are not part of any known classes. Instead, this approach is a distance-based method. In essence, the output of our model is a feature space instead of the logit space. This space has low dimensions and is constructed

to fulfill the following requirements: samples of the same class lie near each other, while samples of different classes are far from each other.

Based on experimental results, we found that it is advantageous to have fixed class centers for using pairwise distances, and the model cannot reliably accumulate all the samples of a class into one area of the feature space. The experiments also showed the necessity of using generated data to represent the intraclass space to optimize the parameters; otherwise, the model contracts the whole input space around the class centers. In this way, the open-set recognition (OSR) problem becomes determining to how to create the appropriate synthetic data samples for the training of the model.

However, generating fake inputs of appropriate quality is a challenging task for a reason more deeply explained in this paper. Moreover, creating and using these samples increases computational costs at two stages of the procedure: First, one must train the generative model, which is computationally expensive if the goal is to create high-quality data. Second, the generated samples must also be run through the classifier.

We thus generated fake features using the features of real data in a hidden layer of an artificial neural network to consistently achieve better results. As the samples classified in this study are image datasets, the backbone of the model is a convolutional neural network. It is built from several convolutional layers, followed by a few fully connected ones. The experiments showed that after all the convolutional layers, generating the features in a hidden layer is optimal in the later components of the model. The much simpler structure of this layer enables the use of a significantly smaller generative model. At the same time, the generated samples have to be run only through the second, a minor part of the classifier. In this way, the further computational costs caused by using synthetic samples are almost entirely eliminated.

The paper is structured as follows: After a brief overview of the related work, we introduce our approach in detail and present the experimental performance results by comparing them to the performance of other existing methods in various OSR scenarios. A detailed analysis of runtimes is also provided, comparing the phases of the model and the models generating features and images.

2. Literature Review

In this section, we briefly review the corresponding literature, beginning with the theory of open-set recognition, then continuing with the most relevant solutions.

2.1. Theory of Open-Set Recognition

Many algorithms have long been used to solve classification tasks where only some samples belong to any known class [1,3] or the machine needs to be more confident [4,5] in classifying. The formalization of the theory of open-set recognition was introduced by Scheirer et al. [6]. We follow their definitions.

Let O denote the open space (i.e., the space far from any known data). The open space Rrsk is defined as follows

$$R_O(f) = \frac{\int_O f(x)dx}{\int_{S_O} f(x)dx} \tag{1}$$

where S_O denotes the space containing both the positive training examples and the positively labeled open space, and f is the recognition function, with $f(x) = 1$ if the sample x is recognized as as a known class and $f(x) = 0$ otherwise.

Definition 1. *Open-Set Recognition Problem: Let V be the set of training samples, R_O be the open space risk, and R_ϵ be the empirical risk (i.e., the closed set classification risk, associated with misclassifications). Then, open-set recognition is the task of finding an $f \in H$ measurable recognition function, where $f(x) > 0$ means classification into a known class, and f minimizes the open-set risk:*

$$\arg\min_{f \in H} \{R_O(f) + \lambda_r R_\epsilon(f(V))\} \tag{2}$$

where λ_r is a regularization parameter balancing open space risk and empirical risk.

Definition 2. *The openness of an open-set recognition problem is defined as follows (according to [6]):*

$$O = 1 - \sqrt{\frac{2 \times |C_{TR}|}{|C_{TA}| + |C_{TE}|}} \tag{3}$$

where C_{TR}, C_{TA}, and C_{TE} denote the training, target, and test classes, respectively.

Extreme Value Theory

SVMs and SoftMax classifiers were initially designed for the closed-set scenario and have been modified to reject open-set samples to some extent. Another example is OpenMax by Bendale et al. [7], who changed a neural network classifier after being first trained in the traditional closed set scenario with a SoftMax layer. However, to achieve better results, the principle of dividing the input/feature space with hyperplanes or otherwise has to be revised, and fundamentally different approaches are needed.

The probability of a sample belonging to a given class can be statistically approached by considering the distribution of distances from the training samples in feature space.

Jain et al. [8] set a solid theoretical base, extreme value theory (EVT), showing the connection between the probability of a sample belonging to a class and the distance of the sample from the known (training samples). Extreme value machine (EVM) directly applies the margin distribution theorem. After fitting a Weibull distribution on each training instance in the feature space, the model is reduced: instances covered by other instances with high probability are discarded. The remaining ones become extreme vectors. EVM is a computationally cheap and fast model. However, extracting appropriate features from the data is not a part of it, which is a severe limitation. The features have to be extracted before applying the model. Descriptors have been used for the tests on image datasets, but these are inferior to the state-of-the-art convolutional networks. Additionally, using these convolutional networks nullifies this model's main advantage: its extremely low complexity.

Extreme value theory is used in various other open-set recognition methods. Jain et al. [9] introduced P_I-SVM to estimate the posterior probability of class inclusion. They used EVT on the pretrained 1-vs-Rest RBF SVM output values to obtain the unnormalized posterior probability. The modification, which Bendale et al. used to create OpenMax, is also based on EVT [7].

2.2. Distance-Based Methods

Distance-based methods inherently fit into the open-set scenario. In addition to deciding which class is the most similar to the sample in question, they provide a value on the extent of the similarity. Using this value, e.g., applying a threshold on it, one can decide whether the sample belongs to the most similar class or is unknown.

Júnior et al. [10] extended the nearest neighbor classifier to the open-set scenario. To decide where sample s belongs, its nearest neighbor t is first taken, then the nearest neighbor u s.t. u and t are of different classes. If the ratio of the distances $R = d(t,s)/d(u,s)$ is less than a threshold T, s is classified with the same label as t; otherwise, it is rejected as unknown.

Instead of using the distances between individual instances, Miller et al. [11] used predefined (so-called anchored) class means. A network projects each input into the logit space. Then, the decision is made according to the Euclidean distances between the logit vectors and the class means.

Sample Generation

The biggest issue with open-set recognition is that the model cannot be trained, and samples have to be rejected, as unknown classes occur only during tests. Using generated data to model these samples can improve the performance of an open-set classifier.

Generative adversarial network (GAN) was created by Goodfellow et al. [12]. It is a method capable of generating images similar to the real ones. It is made of two separate neural networks. One is trained to generate fake data from the noise to deceive the other; the other is trained to separate the actual images from those generated by the first network.

Kong and Ramanan [13] created a version of GAN that generates fake images that is capable of improving the performance of OSR. They supplemented the GAN model with feature vectors for conditioning. Our solution—namely, that we generate inner features instead of images—does not require additional conditioning, making both the training of the GAN model and using the generated samples computationally much more efficient while equallycontributing to the performance of the OSR model. Other solutions, such as those of Jo et al. or Ge et al. [14,15], have used GANs as well. Another method to generate samples is the usage of autoencoders [16]. In one solution [17], autoencoders were used for the classification itself, with the reconstruction error being the base of the decision. Generating negative samples can significantly improve the performance of an OSR model. However, their significant extra computational cost is a major drawback.

3. Proposed Novel Approach

Neural-network-based methods are the most popular and some of the most successful tools for classification tasks. However, adapting them to the open-set scenario is still challenging, considering the SoftMax layer used on these networks. We present a different approach. Instead of applying a SoftMax layer (and thus dividing the state space with hyperplanes), the output of the network is a space in which the distance $d(f(x), f(y))$ is minimal if the samples x and y belong to the same class and are maximal otherwise. $f(x)$ is the output of the network on the input x. Algorithm 1 shows the proposed model. We are describing its details below.

3.1. Fixed Class Centers

It is simpler to use fixed class centers instead of pairwise distances, as was reported in [11]. Hence, there is no need to either explicitly motivate the different classes to be far from each other or to calculate the pairwise distances; the training is thus simplified into a quadratic regression: let $Y = (y_1, y_2, \ldots, y_k)$ be the class centers for the k known classes. Then, the loss function for $(f(x))$, where x is a sample from class i, is simply $||(f(x) - y_i)||_2^2$. The class centres are one-hot vectors and their opposites: $y_1 = (1, 0, \ldots)$, $y_2 = (-1, 0, \ldots)$, $y_3 = (0, 1, 0, \ldots)$, etc.; the dimension of the output space is thus $k/2$. Using the opposites of one-hot vectors makes the origin the average output and, statistically, the expected value when encountering new samples.

3.2. Sample Generation

To improve the ability of the model to recognize unknown inputs, synthetic data are used to model the space of these unknowns during training. The samples are obtained by a generative adversarial network [12]. The samples to train it are positive training inputs. However, the experiments showed that generating appropriate fake images is a challenging task. Using a too-simple model results in low-quality images. Although the model can easily be trained to reject them, training does not sufficiently prepare the model to reject real unknown samples. With a more complex model, we could generate high-quality images; however, training the model to reject these led to the rejection of unseen test samples of known classes. Figure 1 illustrates the two extreme cases. We were unable to reliably find the optimal middle ground.

(**a**) (**b**)

Figure 1. Using a too-simple generator model (**a**), the open-set classifier model does not learn to reject real unknown samples by rejecting these fake ones. However, a complex model generates too-realistic images (**b**), possibly even fooling humans, and the classifier cannot learn to reject these while correctly classifying positive samples. (**a**) Fake images generated by a simple model. (**b**) Fake images generated by a complex model.

Hence, the synthetic samples are generated in a hidden feature layer instead of the input space. This requires the real features from this layer. These are accessible by dividing the neural network model into two parts: N_1 and N_2. N_1 is responsible for the feature extraction, and N_2 transforms the features into the output space. Let f_1 be the function implemented by N_1; similarly, let f_2 be the function implemented by N_2. Then, feeding N_2 with the outputs of N_1 and training the system with the above-described loss function gives $f(x) = f_2(f_1(x))$. Extensive experiments were conducted across various datasets to find the best cutting point of the model. Through these experiments, we empirically proved that cutting the model after all the convolutional layers provides us with the optimal middle ground mentioned before.

Using this setup, the training process is as follows: First, the networks are pretrained on the set of real images $X = (x_1, x_2 \ldots)$ to implement $f(x) = f_2(f_1(x))$. Then, the features $(f_1(x_1), f_1(x_2), \ldots)$ are saved. These features are used to train the generative model and to further train N_2 on these features and on the synthetic ones. *N_1 is not trained anymore or used during the training anymore.* This greatly decreases the computational costs of the second phase of the training in light of this being part of the model with the convolutional layers; thus, this is the most expensive one to run. The features $(f_1(x_1), f_1(x_2), \ldots)$ are utilized as real data to train the generator N_G and discriminator N_D models using the algorithm described by [12]. After training, N_D creates the synthetic features $X_G = (X_G1, X_G2, \ldots])$ from random noise z of the same distribution $p(z)$ used by training the GAN. These fake features are then used alongside the real features $(f_1(x_1), f_1(x_2), \ldots)$ to further train N_2. The loss function is the same for the known features: $||(f(x) - y_i)||_2^2$, where y_i is the predefined class center of class i of sample x. The goal of regression for the fake features is the origin; the loss function is thus $|wf(x)||_2^2$. Figure 2 shows a schematic representation of the algorithm.

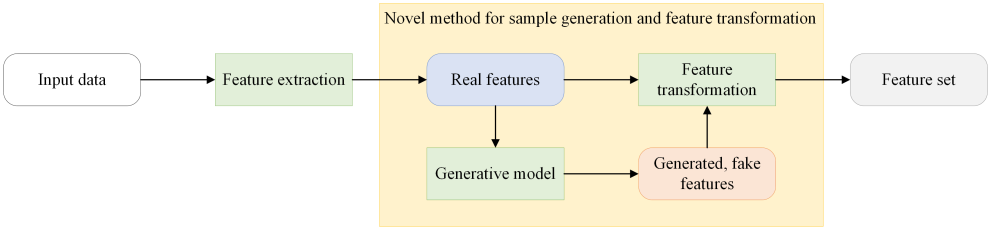

Figure 2. Schematic representation of the model.

4. Experimental Results

We evaluated the performance of our method by comparing it to other relevant methods and a baseline, which was equivalent to our model but without the use of generated negative inputs.

4.1. Implementation Details

As the backbone of the model, different-sized versions of VGGNet were used depending on the size and complexity of the images in the datasets [18]. The convolutional layers, including the max pooling after the last convolutional layer, form the first part of the model. The second part follows the fully connected layers of the same network, e.g., when using VGG19 as the backbone. The second part consists of three hidden layers of 4096, 4096, and 1000 neurons with ReLU as the activation function. The output is a linear layer with several neurons equal to half of the training classes.

Using the generative model in the feature layers enables both the Generator and the Discriminator to be simple, fully connected networks. The Generator consists of two 256-neuron and two 512-neuron linear hidden layers. The input layer has a dimension of 128, i.e., the Generator takes 128-dimensional random vectors as input. The dimension of the output is equal to the dimension of the output of f_1. The Discriminator is symmetrical to the Generator: it has two 512-neuron and two 256-neuron hidden layers, this time completed with LeakyReLU, and the output is a single neuron with sigmoid activation. For training, an Adam optimizer [19] was used with a base learning rate of 3×10^{-4}, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and a batch size of 64.

4.2. Datasets

We compared some of the most common datasets and experimental setups listed below.

- The Street View House Numbers (SVHN) dataset consists of 32×32 images of digits cropped from house number plates [20]. It is considered an easy-to-classify dataset: the accuracy of the best-performing models in closed-set scenarios approaches 99%.
- The CIFAR-10 (Canadian Institute For Advanced Research)[21] dataset, on the other hand, is slightly more challenging. However, it has the same size in terms of individual images and the number of classes. It consists of 32×32 natural images of ten classes, six of which are different animals, with the remaining four being various vehicles.
- For both CIFAR-10 and SVHN, six out of ten classes served as known classes with positive samples; the remaining four provided the unknown samples. The openness of these problems is 13.4%, following Equation (3). The backbone of the model was VGG13 in the case of CIFAR-10 and VGG11 in the case of SVHN.
- TinyImageNet is a dataset similar to ImageNet; one can consider it a downscaled version. The images are smaller (64×64) and contain 200 classes with 500 training images each. The model was trained on only 20 classes from this dataset, and the remaining 180 were used as unknown, resulting in an openness value of 57.4%. This time, the backbone network was VGG16.
- CIFAR+10 and CIFAR+50 involved training the model on four classes from CIFAR-10 and using 10 and 50 nonoverlapping classes, respectively, from CIFAR-100 as unknowns, with the openness of 33.3% and 62.9%, respectively, using VGG13 as in the CIFAR-10 6 + 4 scenario.

In addition to the previously described open-set scenarios, a test on outlier detection was also conducted, following the protocol defined by [22]: we trained the model on all ten classes of CIFAR-10 and tested it on outliers from the ImageNet and LSUN datasets obtained by [23]

Each test was repeated ten times, and the results were averaged, although the deviation of the values was negligible. Random class splits were used (in the case of CIFAR+10 and CIFAR+50 with the described constraints), except for the test on outliers (Table 1), where the protocol strictly specifies the classes used for training.

Table 1. F1 scores on 11 classes (10 known and unknown) tested on various outliers.

	ImageNet-Resize	**ImageNet-Crop**	**LSUN-Resize**	**LSUN-Crop**
Softmax [22]	0.653	0.639	0.647	0.642
Openmax [7]	0.684	0.660	0.668	0.657
LadderNet+Softmax [22]	0.646	0.640	0.647	0.644
LadderNet+Openmax [22]	0.670	0.653	0.659	0.652
DHRNet+Softmax [22]	0.649	0.645	0.649	0.650
DHRNet+Openmax [22]	0.675	0.655	0.664	0.656
CROSR [22]	0.735	0.721	0.749	0.720
C2AE [17]	0.826	0.837	0.801	0.783
CPGM-VAE [16]	0.832	0.840	0.812	0.806
CPGM-AAE [16]	0.830	0.793	0.865	0.820
Baseline	0.773	0.771	0.791	0.766
Proposed method	0.843	0.820	0.819	0.825
Standard deviation	0.016	0.018	0.007	0.011

We also analyzed time complexity, comparing the runtime of a batch on the two parts of the model and the time needed for an iteration on a batch with our generative model and with one that would generate synthetic images.

4.3. Evaluation Metrics

According to a thorough survey on OSR methods by Geng et al., the most common metrics for evaluating open-set performance are AUC and F1 measure [24]. In terms of overall accuracy or F1 measure, the metric is highly sensitive to its calibration, in addition to the real effectiveness of a model [25]. Hence, we evaluated open-set recognition performance with the two metrics described below. The F1 measure was still used to assess performance on the above-described outlier detection, as the other results obtained, followed by the protocol, were also published in this metric. In this case, cross-class validation is used to estimate the difference between the optimal thresholds of the training set—where the positive samples are the positive training samples, and the negative samples are the generated ones—and the test set.

AUC: The receiver operating characteristic (ROC) curve is obtained by plotting the true positive rate (sensitivity) against the false positive rating (1 –specificity) at every relevant threshold setting. The area under this curve gives a calibration-free measure of the open-set detection performance [26].

Closed set accuracy: It is essential that the model, while being able to reject unknown samples, retains its closed-set performance. Therefore, the closed-set accuracy on the test samples of known classes was also measured.

4.4. Results

Table 2 shows the open-set detection performance of the baseline, proposed method, and other approaches in terms of AUC value. The ROC curve in the case of TinyImageNet can be seen in Figure 3.

Table 2. Open-set detection performance in terms of AUC value.

	CIFAR-10	CIFAR+10	CIFAR+50	TinyImageNet	SVHN
Counterfactual [27]	0.699	0.838	0.827	0.586	0.910
CAC-Openset [11]	0.803	0.863	0.872	0.772	0.942
SoftMax [28]	0.677	0.816	0.805	0.577	0.886
C2AE [17]	0.711	0.810	0.803	0.748	0.922
OpenMax [7]	0.695	0.817	0.796	0.576	0.894
G-OpenMax [15]	0.675	0.827	0.819	0.580	0.896
CROSR [22]	–	–	–	0.589	0.899
Baseline	0.720	0.780	0.782	0.662	0.774
Proposed method	0.839	0.825	0.823	0.720	0.845
Standard deviation	0.021	0.018	0.020	0.029	0.027

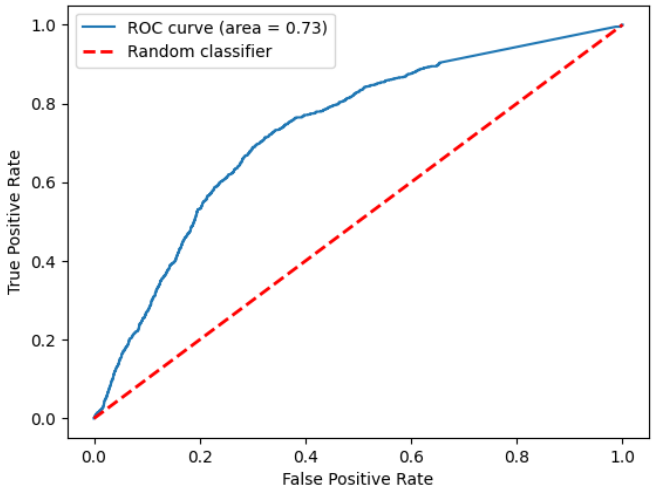

Figure 3. The receiver operating characteristic curve of the model on TinyImageNet in one of the 10 runs.

We employed the same model structure as another comparison base to classify the same inputs. However, without using generated samples, the training was identical to Algorithm 1 to line 10, with the number of iterations n_1 being higher. This method is referred to as a baseline in the tables. Our approach outperforms the baseline and most other listed methods on TinyImageNet, practically tying for first place. At the same time, CIFAR-10 performed the best by a large margin. On SVHN, however, it performed significantly worse. The pattern is clear: our model achieved the best relative performance on the more difficult-to-classify natural images: TinyImageNet is far larger than the others, both in terms of the number of categories and the size of the images. CIFAR-10 is smaller but famously difficult to accurately classify compared with its size. This is exactly the opposite of the result of Neal et al. Their approach worked exceptionally well on SVHN or MNIST. Still, the model's performance significantly dropped on natural images such as CIFAR or TinyImageNet. In the case of CIFAR+ scenarios, the method by [11] outperformed the other methods, but all others produced similar, reasonably good performance, including ours [27].

Algorithm 1 The training procedure of the model

Require: $X = (x_1, \dots x_n)$ training samples, numbers of iterations n_1, n_2
Output: (N_1, N_2, Y) Neural network models and fixed class centers

Initialize N_1, N_2, N_G, N_D with random parameters, class centers $Y = (y_1, \dots y_k)$
$X \leftarrow x$
$N \leftarrow n$
for $i = \{0..n_1\}$ **do**
    for j in batches **do**
        $out \leftarrow N_2(N_1(x_j))$
        $loss \leftarrow quadratic\ loss(out, Y)$
        Update N_1 and N_2 with the gradient of the loss
    end for
end for
$f_1(X) = (f_1(x_1), \dots f_1(x_n) \leftarrow (N_1(x_1)), \dots N_1(x_n))$
$(N_G, N_D) \leftarrow GAN(N_G, N_D, f_1(X))$
$z \leftarrow$ random noise
$X_G \leftarrow N_G(z)$
for $i = \{0..n_2\}$ **do**
    for j in batches **do**
        $out \leftarrow N_2(f_1(X)_j)$
        $loss \leftarrow quadratic\ loss(out, Y)$
        $out \leftarrow N_2((X_G)_j)$
        $loss \leftarrow loss + quadratic\ loss(out, Y)$
        Update N_2 with the gradient of the loss
    end for
end for
return (N_1, N_2, Y)

The closed-set accuracy values are shown in Table 3. No information on the other methods' performance on TinyImageNet was available regarding this measure; hence, we could compare our result only to our baseline. However, our open-set recognition method fully retains its closed-set accuracy; on CIFAR-10 and SVHN, the accuracy even increases when using the generated samples; on CIFAR-10, the difference is significant. Our closed-set accuracy is also superior to that of the other published methods, but with a slight difference on CIFAR-10.

Table 3. Closed-set accuracy across six known classes.

	CIFAR-10	SVHN	TinyImageNet
Counterfactual [27]	0.821	0.951	
OpenMax [7]	0.801	0.947	–
G-OpenMax [15]	0.816	0.951	–
CPGM-VAE [16]	0.912	0.942	–
Plain CNN	0.912	0.944	–
Baseline	0.895	0.952	0.713
Proposed method	0.914	0.964	0.705
Standard deviation	0.032	0.012	0.027

The results of the tests on various outliers are presented in terms of macro-averaged F1 measure (Table 1) to compare them to the results obtained with other methods, which were also published using this metric. However, this metric does not show if the method produces better closed-set accuracy with inferior open-set detection or vice versa. Hence, we considered it proper to analyze and present the AUC and closed-set accuracy as well. The closed-set accuracy was 0.912 , on average, on all ten classes of CIFAR-10, which is similar to the value of 0.914 on the six known classes of the 6+4 open-set scenario. The average

AUC varied between 0.893 and 0.903 across the four categories of outliers. Regarding the F1 measure, our model performed best in two outlier categories, reaching second and third place in the other two, trailing slightly behind the best-performing models. The standard deviation of the results was very low in every case. The nature of the model—the training is simplified to a quadratic regression with fixed class centers—makes it remarkably stable. Only the choice of known or unknown classes slightly varies the results.

4.5. Large-Scale Test on ImageNet

To validate that the model scales well on datasets greater in size and variability, it was tested on the ImageNet (ILSVRC-2012) dataset [29]. Although the existing OSR models were not tested on it by the authors, we could compare the results to the ones on other datasets, for example, the similar TinyImageNet, as well as to the accuracy of the closed-set model VGG19 [18]. For the training, the same network structure was used as for TinyImageNet. The hyperparameters also followed the tests on the other datasets. The results are shown in Table 4.

Table 4. OSR performance on ImageNet regarding AUC and closed-set accuracy with various numbers of known classes used for training. In all cases, every other class provided negative samples. The rightmost column shows the performance of the original VGG19 network on all 1000 classes.

Number of Known Classes	**20**	**40**	**60**	**80**	**100**	**150**	**VGG19**
AUC	0.772	0.761	0.718	0.719	0.730	0.707	–
Closed-set accuracy	0.897	0.844	0.790	0.765	0.793	0.752	0.727

Both the AUC and accuracy values showed only a minor decrease when there were more than 40 training classes, which showed that the model scales well with the size of the problem. Additionally, note that when training on 20 classes, the results were significantly better than those on TinyImageNet, where 20 known classes were used as well, but the images were smaller. This means that the model can utilize the more detail provided by the bigger images of ImageNet.

We also measured the change in the AUC value when varying the number of unknown classes. The model trained on 150 known classes was used. The results are shown in Table 5.

Table 5. AUC performance on ImageNet trained on 150 known classes and tested on various unknown classes. The classes were randomly chosen ten times, except when all of them were used (rightmost column).

Number of Unknown Classes	**100**	**200**	**300**	**500**	**700**	**850**
Average AUC	0.715	0.702	0.674	0.715	0.703	0.707
Standard deviation	0.164	0.058	0.064	0.027	0.008	–

The results were approximately the same regardless of the number of unknown classes, with the only deviance being higher with fewer unknown classes. This shows the strength of the AUC measure, as it is independent of the threshold and the ratio of the positive and negative samples. When testing on a subset of unknown classes, the result varied depending on how hard-to-reject classes are chosen, but the expected value was the same as when testing on all the classes.

4.6. Time Complexity Analysis

Generating samples in an inner layer, in addition to producing better performance in terms of accuracy, results in computationally cheaper training than with generating synthetic inputs. This advantage appears at two distinct points. First, during the training, both the generative and discriminative networks are smaller and of a more straightforward structure than what would be needed for generating images. Second, as the generated samples are already in the hidden layer, during the further training of the model, these

samples do not have to be run through the first part of the model. Hence, by saving the output of the model's first part on the real training samples, there is no need to use this first part for the second training phase.

To measure the time-complexity gain, we obtained measures on runtimes. All the measurements were obtained on the same architecture:

- Intel(R) Core(TM) i7-9800X CPU @ 3.80 GHz;
- NVIDIA(R) GeForce RTX(TM) 2080 Ti;
- 128 GB RAM.

To estimate how the time-gain scaled on real-life (most likely higher resolution) datasets, the measurements were obtained on all three primary datasets used for the experiments.

Table 6 shows the runtimes of the training of the different types of GAN-s.

Table 6. Runtime of the generative model's training on one batch in milliseconds, including running both the generative and discriminative models, calculating the loss value, and using a training step. The third row shows the ratio of the runtimes of the two methods.

	CIFAR-10	**SVHN**	**TinyImageNet**
Image generation (ms)	21	21	76
Feature generation (ms)	6.1	4.8	6.9
Reduction(%)	71.0	77.1	90.9

The values are the runtimes of a training step—running the models, calculating the gradients, and adjusting the weights—on a batch of 64 samples, measured in milliseconds. The difference between the training of the generative models is 3–4 fold on the smaller datasets but exceeds an order of magnitude on TinyImageNet. This led to the conclusion that the time gain achieved by our method is more significant on larger, real-life datasets. The experiments showed that no more iterations are needed to generate appropriate features than to generate images of sufficient quality. Hence, these ratios are the actual gain on the generative part of the training.

We measured the time needed to run a batch of 64 samples through the two parts of the model. The results are also shown in milliseconds in Table 7.

Table 7. Runtime of one batch on the models two parts in milliseconds. The fourth row is the reduction from the sum of the runtimes to the runtime of the second part, as an image has to be run through both parts, while a generated feature only needs to be run through the second part.

	CIFAR-10 (VGG 13)	**SVHN (VGG 11)**	**TinyImageNet (VGG 19)**
First part (ms)	1.3	1.1	3.9
Second part (ms)	0.21	0.21	0.37
Sum (ms)	1.51	1.31	4.27
Reduction (%)	87.3	84.0	90.9

The last row is the ratio of the sum of runtimes and the runtime of the second part, as a generated feature has to be run only on the second part, while a generated (or real) image has to be run through the whole model. For the different datasets, differently sized model structures were used, corresponding to the sizes and difficulties of images. This significantly altered the results between the datasets. It is clear that as the size of the problem scales up, the ratio improves, exactly as in the training of the generative model.

The times needed for a whole training step, also including the calculation of gradients and making the training step, with generated features (only the second part of the model) or with images (the whole model) are shown in Table 8.

Table 8. Runtime of the training in milliseconds, including running the models, calculating the gradients, and adjusting the weights.

	CIFAR-10 (VGG 13)	**SVHN (VGG 11)**	**TinyImageNet (VGG 19)**
Whole model (ms)	8.9	7.3	19.6
Second part (ms)	2.2	2.3	2.7
Reduction (%)	75.3	68.5	86.2

The ratios show a similar pattern to the raw runtime of the model parts. However, they are slightly smaller because a training iteration has some operations that do not scale up with the model's size.

During the experiments, it proved to be optimal to have generated samples in an equal number to all real samples. Considering this, using the synthetic features, the whole training required barely more than half as long as using the generated images—the exact ratio was $(7.26 + 1)/(2 \times 7.26)$ on TinyImageNet—even if both the phases by one are as long in terms of iterations as the whole training with generated images.

5. Conclusions

Open-set recognition faces a new, more complex challenge than the closed-set scenario when all classes are known at training time. Modeling the space of possible unknowns by generating artificial samples was proven to help solve this issue. However, creating and using these samples induce further computational costs.

By using generated features in an inner layer of the model, we made progress both in terms of performance and computational efficiency. Apart from beating the baseline—an identical model without using fake features—our approach performed better than most of the other methods, even with this otherwise elementary model. While generalizing to be able to reject open samples, the model also retains its closed-set accuracy. Up to 150 known classes, the accuracy only dropped down to 75.2%. As a comparison, the classical, closed-set VGG19 network performed 72.7% on all 1000 classes of ImageNet. On two out of the three other datasets, our model outperformed the baseline, which is a closed-set model.

As tests on the ImageNet dataset showed, the model scales well with the size of the problem. When the number of training classes was comparable to the known classes of TinyImageNet, the results were better than on TinyImageNet. This means that the model benefited from more detail because ImageNet has bigger images (256×256 instead of 64×64), which showed that the model retained its strength in feature extraction.

Meanwhile, the computational cost induced by generating and using samples is practically eliminated: On TinyImageNet, generating synthetic inputs took 11 times more time than generating features. While using them for training, the difference was more than seven-fold.

One limitation of the model presented in this paper is that the approach was applied to simpler backbone models. Therefore, its full potential has not yet been used. Applying this approach to other models, on the other hand, is a far from trivial task, which makes further research necessary.

Future work includes applying the above-mentioned generative method to the models representing the state of the art. The baseline on which we applied the method is relatively simple and performs moderately; yet, our model was proven competitive. To adapt this approach to other methods, we must find an appropriate feature layer in which the generative model can work. Each case has to be individually considered; no formula will work with every model.

Author Contributions: Conceptualization: A.P.H. and K.T.; methodology: A.P.H.; formal analysis and investigation: A.P.H., L.S.D. and N.A.H.; writing—original draft preparation: A.P.H. and K.T.; writing—review and editing: K.T., L.S.D., and N.A.H.; funding acquisition: K.T. and T.Z.; resources: K.T. and T.Z.; supervision: K.T. and T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the National Research, Development, and Innovation Office through grant TKP2021-NVA-26.

Data Availability Statement: CIFAR datasets, SVHN, and TinyImageNet, are easily accessible and widely used. The outliers used following the protocol defined by [22] have been made publicly available by the authors at the following site: https://github.com/facebookresearch/odin (accessed on 1 December 2021). Our code is available from the following repository: https://github.com/halan5/fake-feature (accessed on 1 December 2021).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bodesheim, P.; Freytag, A.; Rodner, E.; Denzler, J. Local Novelty Detection in Multi-class Recognition Problems. In Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 5–9 January 2015; pp. 813–820. [CrossRef]
2. Wan, L.; Zeiler, M.; Zhang, S.; Cun, Y.L.; Fergus, R. Regularization of Neural Networks using DropConnect. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 17–19 June 2013; Dasgupta, S., McAllester, D., Eds.; PMLR: Atlanta, GA, USA, 2013; Volume 28, pp. 1058–1066.
3. Nguyen, A.; Yosinski, J.; Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 427–436. [CrossRef]
4. Fumera, G.; Roli, F. Support Vector Machines with Embedded Reject Option. In *Pattern Recognition with Support Vector Machines: First International Workshop, SVM 2002 Niagara Falls, Canada, August 10, 2002 Proceedings* ; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, 2002. [CrossRef]
5. Grandvalet, Y.; Rakotomamonjy, A.; Keshet, J.; Canu, S. Support Vector Machines with a Reject Option. *Adv. Neural Inf. Process. Syst.* **2008**, *21*, 537–544.
6. Scheirer, W.J.; Rocha, A.; Sapkota, A.; Boult, T.E. Toward Open Set Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1757–1772. [CrossRef]
7. Bendale, A.; Boult, T. Towards Open Set Deep Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1563–1572. [CrossRef]
8. Rudd, E.M.; Jain, L.P.; Scheirer, W.J.; Boult, T.E. The Extreme Value Machine. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 762–768. [CrossRef]
9. Jain, L.P.; Scheirer, W.J.; Boult, T.E. Multi-class Open Set Recognition Using Probability of Inclusion. In Proceedings of the ECCV, Zurich, Switzerland, 6–12 September 2014.
10. Júnior, P.; Souza, R.; Werneck, R.; Stein, B.; Pazinato, D.; Almeida, W.; Penatti, O.; Torres, R.; Rocha, A. Nearest neighbors distance ratio open-set classifier. *Mach. Learn.* **2017**, *106*, 1–28. [CrossRef]
11. Miller, D.; Sunderhauf, N.; Milford, M.; Dayoub, F. Class Anchor Clustering: A Loss for Distance-Based Open Set Recognition. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Virtual Conference, 5–9 January 2021; pp. 3570–3578.
12. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2*; MIT Press: Cambridge, MA, USA, 2014; pp. 2672–2680.
13. Kong, S.; Ramanan, D. OpenGAN: Open-Set Recognition via Open Data Generation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021.
14. Jo, I.; Kim, J.; Kang, H.; Kim, Y.D.; Choi, S. Open Set Recognition by Regularising Classifier with Fake Data Generated by Generative Adversarial Networks. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 2686–2690. [CrossRef]
15. Ge, Z.; Demyanov, S.; Chen, Z.; Garnavi, R. Generative OpenMax for Multi-Class Open Set Classification. In Proceedings of the British Machine Vision Conference (BMVC), London, UK, 4–7 September 2017; BMVA Press: London, UK, 2017; pp. 42.1–42.12. [CrossRef]
16. Sun, X.; Zhang, C.; Lin, G.; Ling, K. Open Set Recognition with Conditional Probabilistic Generative Models. *arXiv* **2020**, arXiv:2008.05129.
17. Oza, P.; Patel, V. C2AE: Class Conditioned Auto-Encoder for Open-set Recognition In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.

18. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
19. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
20. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A. Reading Digits in Natural Images with Unsupervised Feature Learning. In Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, 12–17 December 2011.
21. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. 2009. pp. 32–33. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 1 December 2021) .
22. Yoshihashi, R.; Shao, W.; Rei, K.; You, S.; Iida, M.; Naemura, T. Classification-Reconstruction Learning for Open-Set Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4011–4020. [CrossRef]
23. Liang, S.; Li, Y.; Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, 30 April–3 May 2018.
24. Geng, C.; Huang, S.j.; Chen, S. Recent Advances in Open Set Recognition: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 3614–3631. [CrossRef]
25. van Rijsbergen, C.J. *Information Retrieval*; Butterworth–Heinemann: Washington, MA, USA, 1979.
26. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]
27. Neal, L.; Olson, M.; Fern, X.; Wong, W.K.; Li, F. Open Set Learning with Counterfactual Images. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
28. Hendrycks, D.; Gimpel, K. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *arXiv* **2016**, arXiv:1610.02136.
29. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]