Machine Learning with Applications 19 (2025) 100621

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Ensembles of deep one-class classifiers for multi-class image classification

Alexander Novotny [a], George Bebis [b,*], Alireza Tavakkoli [b], Mircea Nicolescu [b]

[a] Department of Computer Science, Virginia Tech, USA
[b] Department of Computer Science and Engineering, University of Nevada, Reno, USA

ARTICLE INFO

Keywords:
Ensembles of classifiers
One-class classification
Multi-class classification
Generative adversarial networks
Auto-encoder
Principal component analysis

ABSTRACT

Traditional methods for multi-class classification (MCC) involve using a monolithic feature extractor and classifier trained on data from all the classes simultaneously. These methods are dependent on the number and types of classes and are therefore rigid against changes to the class structure. For instance, if the number of classes needs to be modified or new training data becomes available, retraining would be required for optimum classification performance. Moreover, these classifiers can become biased toward classes with a large data imbalance. An alternative, more attractive framework is to consider an ensemble of one-class classifiers (EOCC) where each one-class classifier (OCC) is trained with data from a single class only, without using any information from the other classes. Although this framework has not yet systematically matched or surpassed the performance of traditional MCC approaches, it deserves further investigation for several reasons. First, it provides a more flexible framework for handling changes in class structure compared to the traditional MCC approach. Second, it is less biased toward classes with large data imbalances compared to the multi-class classification approach. Finally, each OCC can be separately optimized depending on the characteristics of the class it represents. In this paper, we have performed extensive experiments to evaluate EOCC for MCC using traditional OCCs based on Principal Component Analysis (PCA) and Auto-encoders (AE) as well as newly proposed OCCs based on Generative Adversarial Networks (GANs). Moreover, we have compared the performance of EOCC with traditional multi-class DL classifiers including VGG-19, Resnet and EfficientNet. Two different datasets were used in our experiments: (i) a subset from the Plant Village dataset plant disease dataset with high variance in the number of classes and amount of data in each class, and (ii) an Alzheimer's disease dataset with low amounts of data and a large imbalance in data between classes. Our results show that the GAN-based EOCC outperform previous EOCC approaches and improve the performance gap with traditional MCC approaches.

1. Introduction

Multi-class classification (MCC) is a supervised learning task that involves training a classifier to assign data to one of C different classes where $C \geq 2$; the special cases of $C = 2$ and $C = 1$ are referred to as binary (or two-class) and unary (or one-class) classification problems correspondingly. Deep Learning (DL) methods have demonstrated impressive performance in the context of MCC (Krizhevsky, Sutskever, & Hinton, 2017). Traditionally, MCC methods employ a single monolithic model that simultaneously extracts and classifies features for each class. This approach, however, does not provide flexibility when the class structure of the problem needs to be modified (i.e., by adding or removing classes and/or by splitting or merging classes). Many practical applications require this kind of flexibility such as open set problems (Scheirer, de Rezende Rocha, Sapkota, & Boult, 2013). For example, a system for classifying plant diseases classification might need to be modified by adding new diseases or combining known diseases into the same disease. In this case, the model needs to be retrained even for small changes to ensure optimum performance. Moreover, monolithic multi-class classifiers face challenges when dealing with imbalanced data, where classes are not represented equally (e.g., some classes might have significantly more instances than others due to skewed distributions). This can become an even more serious issue in incremental learning scenarios where models need to continuously learn as new data comes along.

An alternative approach to using a single classifier for MCC is employing an ensemble of binary classifiers (Galar, Fernández, Barrenechea, Bustince, & Herrera, 2011). This requires splitting the MCC problem into multiple, binary classification sub-problems and learning a binary classifier for each. There are two main strategies in this context: $One-vs-All$ and $One-vs-One$. In the $One-vs-All$ strategy,

* Corresponding author.
E-mail addresses: anovotny@vt.edu (A. Novotny), bebis@unr.edu (G. Bebis), tavakkol@unr.edu (A. Tavakkoli), mircea@unr.edu (M. Nicolescu).

https://doi.org/10.1016/j.mlwa.2025.100621
Received 8 December 2024; Received in revised form 29 December 2024; Accepted 8 January 2025
Available online 22 January 2025

a binary classifier is trained on a single class versus all other classes. Assuming C classes, this strategy requires creating an ensemble of C binary classifiers; we typically refer to this type of binary classifiers as $class-specific\ classifiers$ (CSCs) since their task is to identify data samples that belong to a specific category. The final classification is performed by selecting the class with the highest probability. In the $One-vs-One$ strategy, each classifier is trained on a pair of classes. Assuming C classes, this strategy requires building an ensemble of $C(C-1)/2$ binary classifiers, which is significantly higher than the number of classifiers required by the $One-vs-All$ approach. The final classification is performed by aggregating the predictions from each classifier (e.g., using the majority rule). These methods can be generalized by exploiting more diverse binary partitions using error-correcting output coding which trades redundancy for robustness (Dietterich & Bakiri, 1995). Although these strategies can deal efficiently with the issue of removing or merging classes, they still face challenges when adding or splitting classes since retraining some or all of the ensemble classifiers is required. Moreover, they suffer from ambiguous classification regions (Duda, Hart, & Stork, 2000) while the issue of imbalanced datasets could become even more challenging in the case of CSCs.

Addressing the above issues requires designing ensembles of classifiers (ECs) that satisfy two main requirements: (1) each class is associated with a classifier that learns from that class only without any information from the other classes and (2) the rule for aggregating the responses of the classifiers for the final decision can easily accommodate changes in the class structure without requiring extensive modifications or retraining. In the rest of the paper, we will refer to them as $EC\ requirements$. Employing one-class classifiers (OCCs) to design ensembles that satisfy the EC requirements represents a promising research direction. Traditionally, OCCs have been deployed in $anomaly$ detection (Chandola, Banerjee, & Kumar, 2009; Salehi et al., 2022) where the goal is to identify instances that do not fit in well with the rest of the data in a target distribution (e.g., detecting abnormalities in medical images). Our interest in this work is to leverage the attractive properties of OCC to design ensembles of one-class classifiers (EOCC) for solving MCC problems.

One-class classification (Khan & Madden, 2014; Perera, Oza, & Patel, 2021; Seliya, Zadeh, & Khoshgoftaar, 2021) aims to determine whether data samples belong to a target class which is the only class that the classifier has seen during training. This is fundamentally different from class-specific classification where a binary classifier is trained to separate a given class from the rest (i.e., the classifier is trained with data from all classes). Typically, a unary classifier is trained for one-class classification; however, one-class classification can also be addressed using a binary classifier where training uses both target and pseudo-non-target data which is generated without any knowledge of the non-target distribution (Hempstalk, Frank, & Witten, 2008; Kang, 2022; Oza & Patel, 2019). We have investigated both unary and binary classifiers for one-class classification in this work. Once trained, an OCC yields a $confidence\ score$ that determines how close a given sample is to the target class. This confidence score can then be thresholded to perform one-class classification. An EOCC can be used for MCC by first generating a confidence score for each class and then combining them using some aggregation rule (Hadjadji, Chibani, & Guerbai, 2017) provided that it satisfies the second EC requirement.

Using EOCC for MCC provides a flexible framework. Assuming that each class is represented by a single classifier, a total of C classifiers is required for a C-class classification problem. New classes can be added by simply adding new classifiers while existing classes can be removed by removing their corresponding classifiers. Associating an OCC with each class can better address the issue of imbalanced data since each classifier is only trained with data from a single class (Krawczyk, Woźniak, & Herrera, 2015; Lee & Cho, 2006). In particular, EOCC can easily accommodate incremental learning applications where new data is leveraged to further improve classification performance by fine-tuning a subset of OCCs instead of the whole model. They can also better support the "fitted learning" framework introduced by Kardan and Stanley (2018). Moreover, each classifier can be independently optimized, for example, by choosing different features (Baggenstoss, 2004; Tax & Duin, 2001) or models (Kang, Cho, & Kang, 2015). On the other hand, OCCs cannot leverage inter-class information which might be important for some classification tasks. This can be partially addressed using a binary classifier by introducing pseudo-non-target data as discussed earlier.

Using EOCC for MCC has received rather limited attention in the literature, most likely because this approach has not yet matched or exceeded the performance of monolithic multi-class classifiers. In fact, many previous studies used artificial or less interesting datasets (e.g., non-image data from the UCI repository Blake & Merz, 1998) and did not explicitly compare the performance of EOCC with traditional MCC approaches. Additionally, they did not perform any experiments to demonstrate how EOCC handle changes in the structure of the classes (e.g., adding or removing classes). This study benchmarks several classical OCCs based on Principal Component Analysis (PCA) and Auto-encoders (AEs) for designing EOCC to solve MCC problems. Moreover, we leverage recent advances in DL to design more powerful EOCC by proposing some new OCCs based on Generative Adversarial Networks (GANs). AEs, for example, suffer from training stability problems if made too deep, and are therefore difficult to get sophisticated enough for classification tasks. We have performed extensive experiments and comparisons to evaluate the performance of EOCC using more complex computer vision datasets. In our experiments, we have explicitly compared the performance of EOCC with some state-of-the-art DL multi-class classifiers. We have also investigated scenarios where new classes are added or existing classes are removed to better understand the overall benefits of EOCC approaches. It should be mentioned that EOCC can be extended to include a reject option (Tax & Duin, 2008); however, this is beyond the scope of this work.

The rest of the paper is organized as follows: Section 2 provides a brief overview of OCCs and EOCC with emphasis on MCC. Section 3 gives an overview of the datasets used in our experiments for testing the models presented. Section 4 provides an overview of previous methods and introduces new ones, based on DL models, for building EOCC for MCC Section 5 presents our experiments, results, and comparisons. Finally, Section 6 presents our conclusions and directions for future research.

2. Background

This section provides a brief review of OCCs and applications followed by a review of EOCC with emphasis on MCC. OCCs are designed to learn the characteristics of a single class so that they can identify instances that do not belong to that class (Khan & Madden, 2014; Perera et al., 2021; Seliya et al., 2021). This is useful in anomaly detection and related problems such as novelty detection, and out-of-distribution detection (Chandola et al., 2009; Salehi et al., 2022). Anomalous data can be defined as rare and unusual examples that do not fit in with the rest of the data (i.e., outliers). Examples include structural defects, malicious actions like bank fraud and cyberattacks, and abnormalities in medical data. Novelty data represents unknown data that a classifier has not been exposed to during training (e.g., due to ill sampling) but are not in principle anomalies or outliers. Out-of-distribution data represent instances that belong to a distribution different from the target distribution. Depending on the application, OCC designed for anomaly detection could be applied in the context of novelty or out-of-distribution detection. However, these are in general different problems.

One-class classification approaches fall into four main categories: (i) density-based, (ii) boundary-based, (iii) distance-based, and (v) reconstruction-based. Density-based approaches work by fitting a statistical distribution to the target data (i.e., training data representing the target class). Both parametric and non-parametric methods can be

used to model the target class. Parametric methods model the density function using a specific functional form (e.g., Gaussian or mixture of Gaussians) while non-parametric methods use a weighted sum of kernel functions (e.g., Parzen Windows) (Duda et al., 2000). Instances having a low likelihood can then be considered outliers. These methods typically require a large amount of training data for accurate density estimation and their performance depends on selecting an appropriate likelihood threshold.

Boundary-based approaches fit a decision boundary around the target data such that instances falling outside the decision boundary are considered outliers. The main challenge with this approach is how to find the optimal size of the volume enclosing the target class. Some representative methods in this category include One-Class Support Vector Machine (OC-SVM), Support Vector Data Description (SVDD), and One-Class Convolutional Neural Network (OC-CNN). OC-SVM (Scholkopf, Williamson, Smola, Taylor, & Platt, 1999), which is a variation of the Support Vector Machine (SVM) classifier (Hearst, Dumais, Osuna, Platt, & Scholkopf, 1998), finds a hyperplane that separates the training data from the origin while at the same time maximizing the hyperplane's distance from the origin. SVDD (Tax & Duin, 2004) extends OC-SVM by finding a hypersphere with a minimum radius enclosing the target data; any data outside of the hypersphere is considered an outlier. OC-CNN (Oza & Patel, 2019) uses a zero-centered Gaussian noise in the latent space is used as the pseudo-non-target distribution.

Distance-based approaches measure some form of distance to determine how much an unknown instance deviates from the target class; instances that are distant from the training data are rejected as an anomaly. Some representative methods in this category include nearest neighbor (NN) methods (e.g., One-Class Nearest Neighbor (OC-NN) and K-Nearest Neighbor (OC-KNN)), clustering methods (e.g., One-Class K-Means (OC-KM)), Local Outlier Factor (LOF), and Isolation Forest (IF). Giver a test sample, OC-NN (Khan & Madden, 2014) finds its first nearest neighbor in the target class and the first nearest neighbor of this neighbor in the target class. If the ratio of these distances is high, then the test sample is considered an outlier. OC-KNN (Khan & Madden, 2014) is an extension of this idea to using the K nearest neighbors where K is a parameter; the final decision is typically performed using majority voting. Clustering methods such as OC-KM work similarly, however, they consider the distance from the closest clusters instead of the nearest neighbors. LOF (Breunig, Kriegel, Ng, & Sander, 2000), computes the LOF score of a test sample by taking the ratios of the local density of the instance and the local densities of its neighbors; test samples with a high LOF score are considered outliers. IF (Liu, Ting, & Zhou, 2008), which has been inspired by the Random Forests (RF) classifier (Breiman, 2001), builds multiple binary trees that separate the feature space recursively where each node divides its child nodes based on a randomly selected feature and threshold. Instances that are placed deeper in a tree are typically not outliers. The final decision is made by averaging the predictions over all binary trees.

Reconstruction-based methods work by first projecting the data in a low-dimensional manifold which is embedded within a higher-dimensional space formed by the target class. The projected data is then reconstructed back in the original space and the error between the original and reconstructed data is computed. The reconstruction error can be used to estimate the distance of the data from the target class and perform anomaly detection. The motivation is that data that fit well in the target class have a low reconstruction error while other data have a higher reconstruction error. A popular one-class classifier based on this idea was introduced in the context of face detection by Turk and Pentland (1991) using PCA (Duda et al., 2000). Since PCA can only compute a linear manifold, extensions to the case of non-linear manifolds have been proposed using Kernel Principal Component Analysis (KPCA) (Scholkopf, Smola, & Muller, 1998) and AEs (Goodfellow, Bengio, & Courville, 2016).

Hybrid OCC methods have also been proposed, for example, using a boundary-based OCC (e.g., SVDD) for pseudo-density one-class classification (Lee & Lee, 2007) or combining boundary-based with density-based OCCs (Hempstalk et al., 2008). OCCs have been extended to EOCC (Khan & Madden, 2014; Perera et al., 2021; Seliya et al., 2021) since the target class might not always be characterized well by a single OCC (i.e., due to multi-modal data); this is analogous to using ensembles of multi-class classifiers (Dietterich, 2000). To avoid confusion, an EOCC used to solve a one-class classification problem will be denoted as OC-EOCC while an EOCC used to solve an MCC problem, which is the focus of this work, will be denoted as MC-EOCC. OCCs and OC-EOCC have been primarily used for anomaly detection (Chandola et al., 2009; Fernando, Gammulle, Denman, Sridharan, & Fookes, 2021; Salehi et al., 2022), out-of-distribution detection (Chandola et al., 2009; Salehi et al., 2022) and novelty detection (Pimentel, Clifton, Clifton, & Tarassenko, 2014; Salehi et al., 2022). Krawczyk et al. (2015) have investigated what type of MCC problems might be most appropriate to solve using EOCC; this includes problems suffering from imbalanced data, overlapping distributions, and high number of classes. We review below some representative MC-EOCC approaches.

An MC-EOCC can be designed using several different strategies: (i) assign the same type of OCC to all the classes (i.e., homogeneous EOCC), (ii) assign different types of OCC to some of the classes (i.e., heterogeneous EOCC), (iii) assign a homogeneous or heterogeneous EOCC to each class instead of a single OCC (i.e., this will be an ensemble of ensembles which will be denoted as EEOCC), and (v) use different aggregation rules to combine the outputs of the classifiers. The diversity of classifiers in an ensemble can be controlled in different ways, for example, by choosing different features, training data, architectures, models, etc. It should be noted that in the case of heterogeneous ensembles, it is imperative to normalize the classifier outputs before aggregating them (e.g., by converting their outputs to pseudo posterior probabilities using softmax normalization (Hadjadji et al., 2017)). In general, the outputs of the classifiers in an ensemble can be aggregated using static (i.e., same aggregation scheme for all test data), dynamic (i.e., adapting the aggregation scheme on the fly based on the test data), or trained rules (i.e., learn the aggregation rule by training a meta-classifier) (Duin, 2022; Kittler, Hatef, Duin, & Matas, 1998). The "max" and "min" rules are commonly used static rules for density/boundary-based and reconstruction-based EOCC respectively.

Ban and Abe (2006) investigated building an MC-EOCC using SVDD and KPCA for one-class classification. A separate classifier was trained for each class and a minimum-distance aggregation rule was used to combine the responses of the OCCs to decide the correct class for MCC. Their MC-EOCC approach resulted in comparable performance to a $One - vs - One$ multi-class classification approach (i.e., using binary SVMs) on several benchmark datasets (most of them from the UCI repository), containing between two and four classes. The authors concluded that MC-EOCC could lead to better generalization performance provided appropriate parameters are chosen. In Lee and Lee (2007), a similar MC-EOCC framework was proposed using SVDD classifiers, however, the responses of the classifiers were aggregated using the Bayes rule by treating them as pseudo-posterior probabilities that were computed by converting the SVDD outputs to pseudo-density estimates. Their results showed comparable performance to six traditional multi-class classifiers on a high number of both small and large-scale datasets. In Hao, Chiang, and Lin (2009), an ensemble of CSCs using SVDD classifiers was proposed where each SVDD classifier was trained using a "One-vs-All" approach to improve the bounding hypersphere of each class. Although the authors reported some performance improvements on various datasets (half of them from the UCI repository) compared to traditional multi-class classifiers, their approach violates the first EC requirement. Garcia, de Sá, Poel, Carvalho, Mendes-Moreira, Cardoso, de Carvalho, and Kok (2021) proposed an MC-EOCC approach for human activity recognition using accelerometer data. Each human activity was assigned to a separate class which was represented by a reconstruction-based OCC based on AEs (i.e., each AE effectively learns to recognize that activity only using the minimum reconstruction error). The same

AE architecture was used for all classes. To demonstrate the efficiency of EOCC in the context of this problem, they experimented with online learning where the system learns incrementally during testing (i.e., only a subset of AEs needs to be updated in this case). Moreover, they experimented with changing the class structure of the problem by combining similar activities into a "super" activity which only required combining the corresponding OCCs. Extensive experiments and comparisons using several benchmark datasets showed that their approach was competitive with conventional methods while being more modular and robust to data from different users.

Several studies have considered improving the performance of MC-EOCC methods by using MC-EEOCC. Krawczyk and Woźniak (2014) introduced a homogeneous MC-EEOCC approach for classifying breast cytological (i.e., biopsy) images into three categories: benign, malignant, and fibroadenoma. Each class was represented by an ensemble of SVDD classifiers trained on different sets of features to ensure good diversity. Pruning was performed to remove redundant classifiers. Their fusion scheme was customized for this problem based on input from physicians. The authors evaluated their method on a relatively small dataset reporting higher performance when compared with an ensemble of OC-SVM classifiers and an RF multi-class classifier. In a related application, Zhang, Zhang, Coenen, Xiao, and Lu (2014) proposed a homogeneous MC-EEOCC using KPCA classifiers trained on shape and texture features extracted by different methods. The confidence of each class was computed by aggregating the classifiers associated with the class using a variant of the product rule (Kittler et al., 1998) and choosing the class with the highest confidence. Tests were performed using three medical datasets (one from the UCI repository), containing between two and three classes. Their method demonstrated better or similar performance when compared with classifiers trained on a single type of feature; no comparisons were performed with MCC approaches. Juszczak and Duin (2004) illustrated how to leverage a homogeneous MC-EEOCC to handle missing data. In their approach, each class was represented by multiple Parzen classifiers (Duda et al., 2000) where each classifier was trained with a different feature. During classification, only the available features were classified and the confidence of each class was computed by aggregating the responses of the classifiers associated with that class using a fixed rule; the class with the highest confidence was then chosen for the final classification. Their method was tested using several datasets from the UCI repository and was found to outperform a single classifier trained on all features using traditional techniques to replace the missing features.

Hadjadji et al. (2017) introduced a heterogeneous MC-EEOCC along with a more powerful aggregation rule based on dynamic weighted averaging. In their approach, each class was represented by five heterogeneous OCCs. During classification, the confidence of each class was computed by weighting the classifiers associated with the class dynamically. This was performed by calculating the weight assigned to each classifier for each test sample specifically according to the classifier's maximum output during training. The test sample was then assigned to the class with the highest confidence. Their results on nine benchmark datasets (containing 2–100 classes), showed improved performance when compared to a KNN MCC. Interestingly, the authors reported that aggregating a subset of the five OCCs is sufficient for achieving the best performance for some datasets. Kang et al. (2015) considered a heterogeneous MC-EEOCC approach, however, they employed a trained meta-classifier for aggregating the classifiers' outputs which violates the second EC requirement. Their results were only compared to other MC-EEOC methods. Fragoso, Cavalcanti, Pinheiro, and Oliveira (2021) proposed an MC-EEOCC where the diversity of each EOCC was controlled by dividing the feature space of each class into different clusters and associating a separate OCC with each cluster. Given an unknown instance, the confidence of each class was computed by selecting a subset of the OCCs associated with that class using a dynamic selection scheme. Each class was represented by four OCCs (two density-based, one distance-based, and one boundary-based). Extensive experiments using 25 datasets showed improved performance compared to other MC-EEOCC approaches; however, no comparisons were performed with traditional MCC methods. A similar approach but with different aggregation strategies has been reported in Krawczyk, Wozniak, and Cyganek (2014) and Krawczyk, Galar, Woźniak, Bustince, and Herrera (2018).

Table 1
Properties of the datasets used for training and testing.

	Size (px)	Classes
Alzheimer's	176 × 208	4
Corn		4
Strawberry		2
Potato	255 × 255	3
Apple		4
Tomato		10

3. Datasets

This section reviews the image datasets used in our experiments. An overview of the different datasets is presented in 1.

3.1. Plant village dataset

A subset of the Plant Village dataset (Hughes, Salathé, et al., 2015) was selected in this study. The dataset consists of images of plant leaves from 14 different plants, each with multiple different diseases, with a total of 38 different classes. It comes in color, monochrome, and segmented versions. We used the segmented version, as there were some flaws in the non-segmented color version, which led to obvious dependencies between the backgrounds and plant/disease types. In this study, we chose the following five plant types which contain a total of 27 classes: Corn, Strawberry, Potato, Apple, and Tomato. Examples of the different plants examined in this study can be seen in Fig. 1, and examples of the different diseases for the Tomato plant can be seen in Fig. 2.

Due to the hierarchical nature of this dataset, many different types of classification experiments can be performed, such as classifying the different types of plants, the diseases of a specific plant, or between diseases of different plants. This is an ideal example of a dataset for which someone may want to change the class set structure under examination — such as by adding or removing relevant plants and diseases as the seasons change, or when there is a new outbreak. Also, some prior benchmarks for this dataset lack rigorous results by way of cross-validation, which provides us an opportunity to improve these benchmarks (Atila, Uçar, Akyol, & Uçar, 2021). It should be noted that inside a particular plant's dataset, many of the classes share coarse features — such as the shape and overall colors of the leaf. Where these classes differ is in the fine features, such as small spots on the leaf, which are more difficult to classify in general.

A list of all of the classes examined in this study and their sizes can be found in Table 2. Note that the Potato and Tomato contain the largest outliers in terms of sample size with the Healthy and Yellow Leaf classes, respectively. Since these datasets contain such outliers, we will focus on them when analyzing the effect of class imbalance on model performance.

3.2. Alzheimer's disease dataset

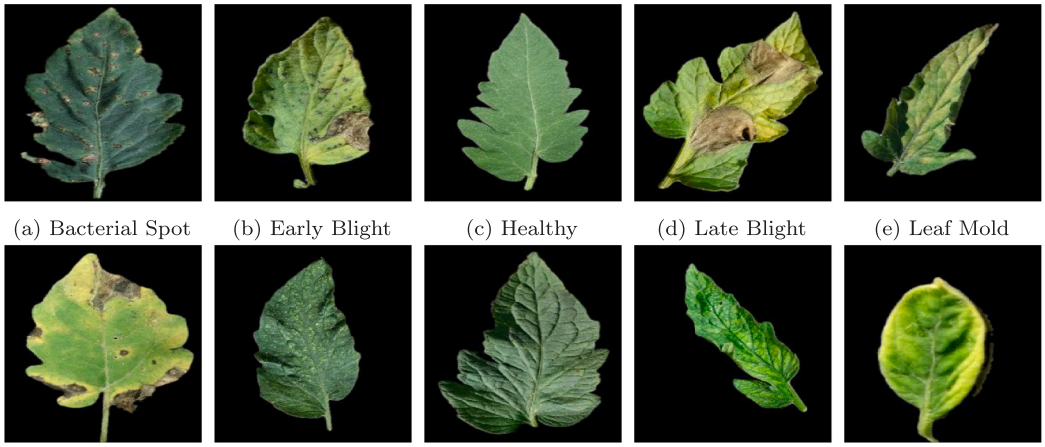
The Alzheimer's dataset was selected in this study to evaluate the performance of the proposed models on complex medical images with very little training data (Dubey, 2019; Yakkundi, 2023). The dataset consists of single monochrome slices of a brain scan in patients with or without Alzheimer's. Examples of images from the different classes of this dataset can be seen in Fig. 3, and a breakdown of sample size by class can be found in Table 3. All images are resized to 256 × 256 using

Table 2
Class breakdown of all of the PlantVillage datasets examined in this study by sample size.

Dataset	Classes & No. samples				
Corn	Cercospora	Common Rust	Healthy	Northern Leaf Blight	
	513	1192	1162	985	
Strawberry	Healthy	Leaf Scorch			
	456	1109			
Potato	Early Blight	Healthy	Late Blight		
	1000	152	1000		
Apple	Apple Scab	Black Rot	Cedar Apple Rust	Healthy	
	630	621	275	1645	
Tomato	Bacterial Spot	Early Blight	Healthy	Late Blight	Leaf Mold
	2127	1000	1591	1909	952
	Septoria Spot	Spider Mites	Target Spot	Mosaic Virus	Yellow Leaf Curl
	1771	1676	1404	373	5357

(a) Corn (maize) (b) Strawberry (c) Potato (d) Apple (e) Tomato

Fig. 1. Examples of some of the different types of plants included in PlantVillage.



(a) Bacterial Spot (b) Early Blight (c) Healthy (d) Late Blight (e) Leaf Mold

(f) Septoria Spot (g) Spider Mites (h) Target Spot (i) Mosaic Virus (j) Yellow Leaf Curl

Fig. 2. Samples from each of the different classes in the Tomato subset of PlantVillage.

Table 3
Class breakdown of the Alzheimer's dataset by sample size.

Class	No. samples
Healthy	2560
Moderate	52
Mild	717
Very Mild	1792

bicubic interpolation for model compatibility. Note that the images of this dataset are relatively more complex than the PlantVillage dataset while having fewer samples outside the Healthy class. Similarly to the single-plant datasets, the differences between classes here lie in the fine features, rather than coarse ones.

4. Ensembles of one-class classifiers

Our work falls into the MC-EOCC category where each class is represented by a single OCC. Each OCC accepts a sample (in our case, an image) as input, and produces a "confidence score" which can be used to determine how close (or how different) is the sample to the class the model was trained on. To construct an ensemble of OCCs, we take an input sample, pass it to all of the OCCs, record the scores, and use a meta-classifier to determine the class from these scores. An overview of this architecture is shown in Fig. 4. In this study, we use a simple meta-classifier, satisfying the second EC requirement, which picks the class associated with the classifier that produced the smallest score (or largest, depending on the method). Some care must be taken, though, as scores produced by some methods may have a different scale depending on the class. For instance, more detailed classes will have more inherent reconstruction errors associated with them than other classes. In this case, the ensemble will be biased away from picking that class. Some attempts were made to account for this, such as by using a meta-classifier which standardizes all scores with their observed mean and standard deviation on a validation set (e.g., subtracting the mean and dividing by the standard deviation), however, we were not able to find a simple method which consistently produced better results than simply

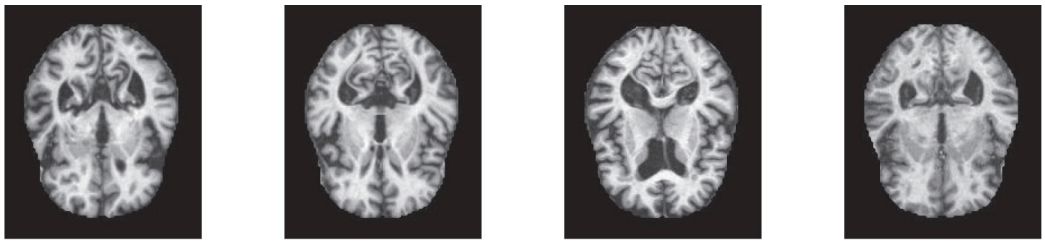


Fig. 3. Samples from each of the different classes in the Alzheimer's dataset.

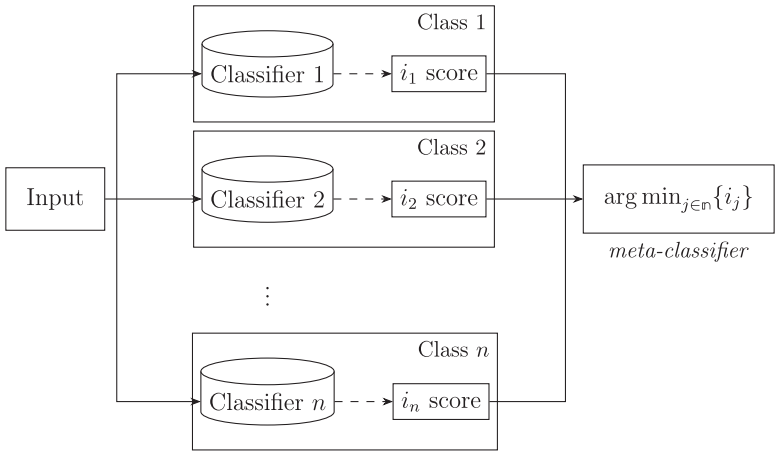


Fig. 4. An overview of the proposed MC-EOCC approach.

picking the lowest/highest score. However, more sophisticated methods are possible at this stage (e.g., adaptive weighting) as described in the previous section. Next, we review the PCA and AE OCCs; then, we introduce the new OCCs based on GANs.

4.1. PCA OCC

PCA was used as an OCC in the seminal work of Turk and Pentland (1991) in the context of face detection (i.e., decide whether an image is a face or not). This is a binary classification problem which can also be solved using an OCC by only modeling the face class. Using PCA, all images in the face class are converted to feature vectors, and their covariance matrix is computed. The eigenvectors of this matrix are then sorted by decreasing eigenvalues, and only the ones corresponding to the largest eigenvectors are kept (referred to as "principal components"). Each of the principal components is associated with a certain amount of information captured by that component which is characterized by its corresponding eigenvalue. Therefore, the first several principal components retain the largest amount of information in the dataset and therefore have high reconstructive power. From a geometric point of view, the principal components define a manifold of lower dimensionality, called PCA space, which is embedded within the original higher dimensional space. We refer to the dimensionality of this manifold, which is determined by the number of principal components, as the latent dimension.

An unknown image can be projected onto the PCA space and then reconstructed back in the original space, which is used to calculate a reconstruction error by taking the magnitude of the difference between the original image and the reconstructed image (see Fig. 5). The further an image is from the span of the principal components (referred to as "distance from face space" in Turk and Pentland (1991)), the larger the reconstruction error. Therefore, the reconstruction error is a good way to determine whether a given image is a face or not which can

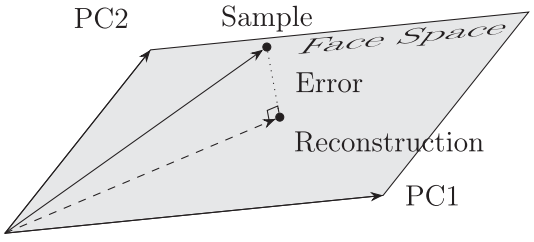


Fig. 5. An illustration of the "face space" constructed by PCA, and where the reconstruction error is obtained from. Principal Components labeled as "PC#".

be used for one-class face detection. The optimum number of principal components to keep is usually determined by experimentation (e.g., using a validation set) and there is not necessarily a direct correlation between overall reconstructive power and discriminative power, since we cannot determine which principal components encode information common among other classes without training on those classes.

4.2. AE OCC

An AE is a neural network which is designed to encode an input into a compressed representation, and then decode it back such that the reconstructed input is as close as possible to the original one. This compressed representation is imposed by a bottleneck layer which in essence defines the latent space where the data is projected; the number of nodes in the bottleneck layer defines the latent dimension. The network is trained in an unsupervised way to minimize the reconstruction error. An AE can be thought of as a generalization of the PCA approach which can find a non-linear manifold assuming non-linear activation functions. Many different types of AEs have been proposed including sparse auto-encoders (SAE), denoising auto-encoders (DAE), variational

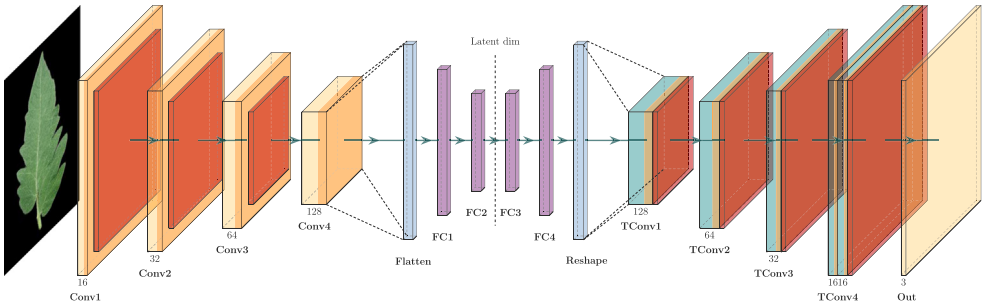


Fig. 6. A visual representation of the proposed AE architecture.

auto-encoders (VAE), and adversarial auto-encoders (AAE) (Goodfellow et al., 2016) with important applications in many different areas including one-class classification and anomaly detection. In this work, we have experimented with deep convolutional AEs.

A visual representation of the AE architecture we have adopted in this study, after some experimentation which was by no means exhaustive, is shown in Fig. 6. The encoder half of the AE architecture is made up of 4 down-sampling blocks. Each down-sampling block consists of a 5×5 convolutional layer with a stride of 2, an activation layer of Leaky ReLU, and a batch normalization layer to improve training (Ioffe & Szegedy, 2015; Maas, Hannun, Ng, et al., 2013). The first down-sampling block has 16 convolution filters, and each block doubles the number of filters, therefore reducing the number of features after each block by half. Convolutional stride was chosen over pooling by validation results. After the down-sampling blocks, the features are flattened, and two dense layers with Leaky ReLU activation follow. The last dense layer has an output the size of the latent dimension (512), while the previous one has the average size between the output of the down-sampling blocks and the latent dimension.

The decoder half of the AE architecture is much the same but in reverse. It begins with 2 dense layers with an output of the same number of features as the down-sampling blocks, which are followed by 4 up-sampling blocks. Each up-sampling block consists of a 5×5 transpose convolution layer with a stride of 2, an activation layer of Leaky ReLU, and a batch normalization layer (Long, Shelhamer, & Darrell, 2015). The first up-sampling block has as many convolutional filters as the last down-sampling blocks, and each up-sampling block halves this, doubling the number of features after each block. There are two more convolutional layers with no stride, Leaky ReLU, and the same number of convolution filters after the up-sampling blocks, and a final convolutional layer with no stride, no activation function, and 3 filters to produce the final resulting image.

An image can be fed into the encoder half of the AE, which will learn to eliminate dependencies between features in the original image and produce a compressed latent representation of the image. This latent representation can then be fed to the decoder half of the AE, which will learn to re-introduce these dependencies and reproduce the original image. Then, the magnitude of the difference between these two images can be construed as a reconstruction error, similar to PCA, and can be used as a confidence score for one-class classification. The smaller the latent dimension, the more dependencies these pieces will have to learn to remove and re-introduce, and the worse the overall reconstruction. A latent dimension which is too large will train into an identity function, reconstructing images from outside the class perfectly and losing its discriminatory ability, while a latent dimension which is too small will fail to capture fine details, where much of the differences between classes lie. Unlike PCA, there is no ordering of latent variables by "importance" - it is unclear which latent variables contribute more significantly to reconstruction than others, and with current losses used to train AE, there is no way to specify such an ordering. Moreover, the loss functions used to train AE typically employ a global reconstruction error (such as in Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE)), which encourages the picked latent variables to encode information about coarse features, since they contribute to more pixels. However, classes with similar coarse features that differ only in fine features (such as with the datasets examined in this study) are more difficult to classify. Similarly to PCA, there is no way to tell which latent variables encode information that is common between classes, and no way to train such a model without having access to other classes' training data.

Attempts to increase the depth of the AE to improve performance, either by increasing the number of up/down sampling blocks or by including more convolutional layers per block as in the U-Net architecture (Goodfellow et al., 2016), were met with difficulties in training the model (Ronneberger, Fischer, & Brox, 2015). It seems like this was due to the very problem that U-Net sought to solve — vanishing gradients. The way U-Net and many other deep networks solve this problem (skip connections) does not necessarily fit our use case — the variables "skipping" across the bottleneck of the AE are actually part of the latent dimension, and therefore we must either greatly enlarge the latent dimension or reduce the capacity of each skip connection.

4.3. GAN inversion OCC

To address the shortcomings of deep AEs, we considered a GAN-based approach (Goodfellow et al., 2016). A GAN is a generative deep neural network that learns how to generate new data that resemble the training data by drawing samples from the distribution of the training data without explicitly modeling it. GANs consist of two separate neural networks, the generator, and the discriminator, which are trained to compete against each other. The generator tries to produce fake data to trick the discriminator, while the discriminator tries to distinguish fake data from real data, forcing the generator to produce more realistic data. From an implementation point of view, the generator is implemented as a decoder since it learns to generate images by drawing samples from a latent space while the discriminator is implemented as a binary classifier.

The main idea of our approach is to design deep AEs using GANs since several techniques are available to train GANs at significant depth, such as progressive growing (Karras, Aila, Laine, & Lehtinen, 2017). Specifically, we first train the decoder part of the AE as the generator of the GAN; then, we train the encoder part of the AE separately using a GAN inversion approach to map images to the latent space variables used to generate them (Creswell & Bharath, 2019; Xia et al., 2022; Zhu, Krähenbühl, Shechtman, & Efros, 2016). Among the different GAN architectures available, we opted for StyleGAN since it uses skip connections from the latent variables to each up-sampling block, rather than skip connections that go across the bottleneck layer like in U-Net (Karras, Laine, & Aila, 2019). These skip connections help training stability as well as enforcing relationships between certain "styles" (as the latent variables are referred to) and coarse/fine features of the resulting images. In this way, some amount of control over specifically coarse or specifically fine features can be exerted by altering the styles differently for the different skip connections.

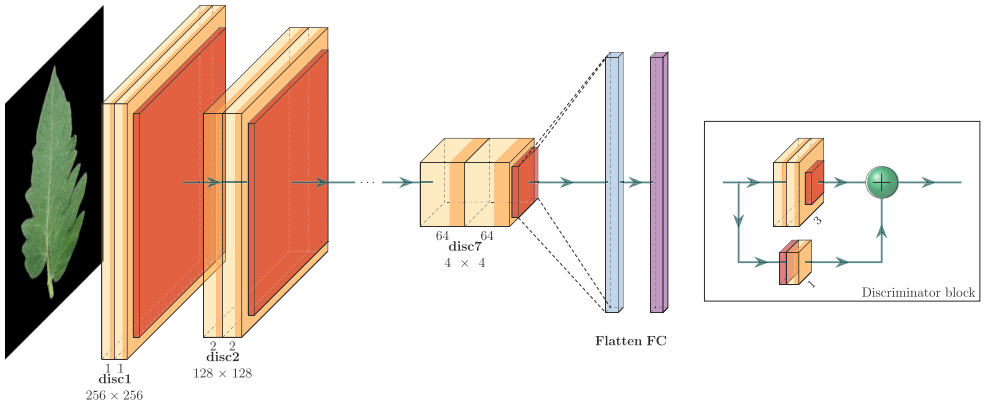


Fig. 7. A visual representation of the proposed inversion network.

The StyleGAN 2 architecture was chosen due to these benefits, and in addition, the increase in inversion performance afforded by the second version, while the texture sticking issue fixed by version 3 of the architecture is unneeded (Karras et al., 2020). We refer to this technique as StyleGAN-I.

To implement the inversion network approach, we used the feature extractor part of the StyleGAN classifier, as suggested by Epstein, Park, Zhang, Shechtman, and Efros (2022). The feature extractor consists of several discriminator blocks, which are composed of two 3×3 convolution layers followed by a $2\times$ bilinear down-sampling layer. A residual skip connection is added from the beginning of the block to the end, which consists of a single down-sampling operation and a 1×1 convolution. Like the encoder part of the AE architecture described in Section 4.2, the resolution is halved in each block, while the number of convolution filters doubles. Due to the inclusion of residual skip connections, there are enough of these blocks to reduce the resolution of the feature maps to 4×4 (7 blocks for the 256×256 images examined in this study) before a flattening and dense layer is used to map the feature maps to the latent variables (512). The dimensionality of the latent space (512) was kept the same as in the AE model for a fair comparison. The training process of StyleGAN-I includes the following steps:

1. For each class, train a generator which produces synthetic data from a latent representation (noise).
2. For each generator, train an inversion network to "invert" an image back to its latent representation,
3. Use the inversion network as an encoder and the generator as a decoder, and continue as in the AE approach (see Fig. 7).

To improve classification results using the inverted StyleGAN, a "jitter" technique was used to emphasize differences in fine features between classes; we refer to this technique as StyleGAN-IJ. Inspired by the sampling process of VAEs (Goodfellow et al., 2016), we introduce some Gaussian noise to the latent variables produced by the inversion network before reconstructing with the StyleGAN generator (Kingma & Welling, 2013). The Gaussian noise is typically introduced to the latent variables at certain resolutions in the up-sampling process of the generator, to emphasize features at specific granularities. Since we have had problems with mean reconstruction error losses over-emphasizing coarse features, the Gaussian noise is only introduced in the final two up-sampling blocks, where fine features are constructed. We sample the noise and reconstruct it multiple times, performing classification with reconstruction loss each time, and then use these as votes. The final classifier picks the class with the most votes over this resampling process. The StyleGAN-IJ methodology is summarized below:

- Before decoding, introduce Gaussian noise to the latent representation.
- Use noisy latent representation for only the fine features (last layers in the generator).
- Using noise resampling, re-evaluate the output of the network several times, and cast a vote each time.

4.4. GAN discriminator OCC

While training a GAN, a fair amount of work is put into training both the generator and discriminator; however, after training, the discriminator is typically discarded and only the generator is used, as in the inversion network approach described above. Here, we are interested in investigating the possibility of directly using the discriminator for one-class classification. The motivation comes from the fact that since the discriminator is trained to differentiate between real and fake images associated with the input distribution, it might be possible that images from another class will be classified as fake by the discriminator. Therefore, we can perform one-class classification by feeding images to the discriminator obtained at the end of the training process and thresholding the output scores. However, as can be seen in Fig. 8, a reasonable discriminator can eventually outpace the generator in the adversarial game. It does this by paying attention to minute differences in real and generated images introduced by the generator which might not even be visible to the human eye and are unlikely to be present in images from other classes. Therefore, real images from other classes might have similar scores to real images from the target class which can also be observed in Fig. 9. We have investigated two variants to address this issue.

The first variant considers discriminators from across the training process instead of the final discriminator. The motivation is that if an image from another class is evaluated by discriminators from across the training process, its scores will vary wildly until becoming more consistent as the training process goes on; this as can be seen in Fig. 10. Under this scenario, we can compute the variance of the discriminator score across all discriminators saved from training to estimate the confidence score for each class. We refer to this variant as "discriminator history" (StyleGAN-DH). The StyleGAN-DH methodology is summarized below:

- Save discriminators with some periodicity while training StyleGAN.
- Evaluate a test image on all of these discriminators and calculate the variance of the scores obtained.
- Use the variance as a confidence score.

Another way to avoid the above issue is to ensure that all images seen by the discriminator are fake. This can be done by using the inversion network described in Section 4.3 to reconstruct all images before showing them to the discriminator. In this way, all images will have the defects introduced by the generator that the discriminator has learned, and the only difference in scores will be from the difference in

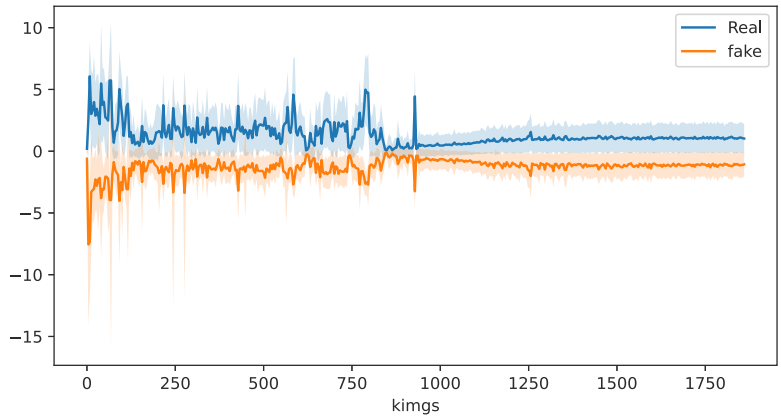


Fig. 8. StyleGAN discriminator scores of real and fake images over training time for the "Tomato healthy" class. Scores are captured as a moving average, with standard deviation shown in shaded regions.

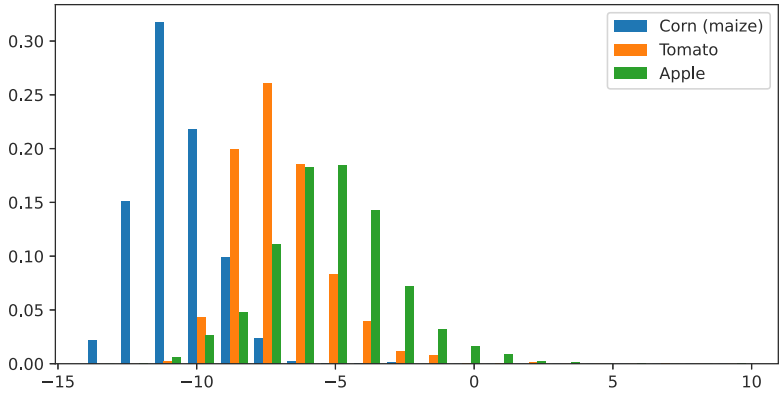


Fig. 9. Histogram of discriminator scores of images from a variety of classes. The discriminator was trained on the "Tomato" plant.

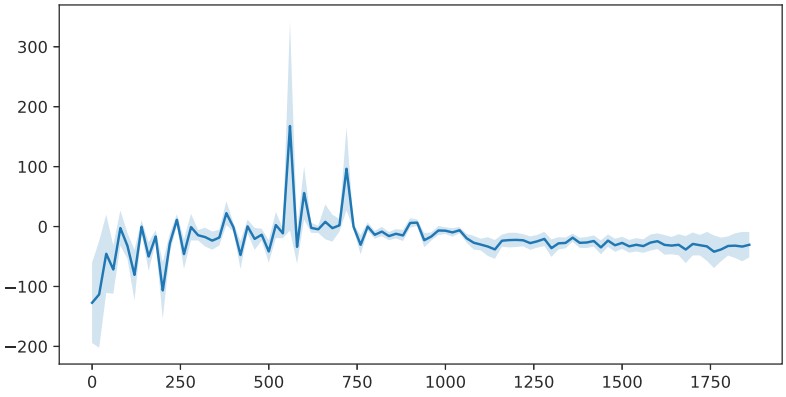


Fig. 10. Mean discriminator scores of Corn plant images evaluated on all discriminators trained on Tomato plant images. Standard deviation is shown in shaded regions.

classes. The discriminator score can be used as a class confidence score although inverted (i.e., the meta-classifier needs to choose the maximum score). We refer to this variant as StyleGAN-ID. The StyleGAN-ID methodology is summarized below:

- Apply inversion network to a test image.
- Reconstruct image with generator.
- Apply discriminator, use discriminator score as confidence score.

5. Results and discussion

5.1. Training

To obtain more rigorous results than previous works with the PlantVillage dataset, a 68%-17%–15% training–validation-test split was used across all datasets. First, the test set was split off of every dataset and kept separate. Then the remaining data was split into 5 equal

Table 4
Mean total training time (i.e., average over 5-fold cross validation) of traditional MCCs on the tomato subset in hours.

VGG-19	Resnet-50	EfficientNet B3	EfficientNet B5
11.38	6.27	10.15	10.47

Table 5
Mean total training time (i.e., average over 5-fold cross validation) of OCCs on the tomato subset in hours.

AE	StyleGAN	Inversion network
2.25	8.79	2.39

17% parts for 5-fold cross-validation, with each 17% part being the validation set in one fold, with the rest of the data left for training. The StyleGAN models were developed and trained using the Pytorch Python framework due to the original work being done using that framework, but the rest of the models were trained using the Tensorflow Python framework (Abadi et al., 2015; Paszke et al., 2019). All models were trained on an Nvidia RTX 3090 GPU with 24 GB of VRAM.

For the PlantVillage datasets, an augmentation layer was applied to all models before input, which included random horizontal and vertical flips, random 90-degree interval rotations, random translations up to 5% of image size, and random zoom up to ±5% of image size. These augmentations were not used for the Alzheimer's dataset, due to non-symmetry and registration of images. All images were standardized to $[-1, 1]$ for training stability.

For benchmarking, several traditional classifier architectures were chosen for comparison. These architectures were VGG-19, Resnet-50, and EfficientNet, which are all well-known architectures in the field of multi-class classification (He, Zhang, Ren, & Sun, 2016; Simonyan & Zisserman, 2014; Tan & Le, 2019). EfficientNet has also some prior work in the PlantVillage dataset (Atila et al., 2021). EfficientNet B3 and B5 were chosen to demonstrate the differences between EfficientNet architectures and because of the similar number of parameters to Resnet-50. Each traditional classifier was trained on each fold for 120 epochs with categorical cross-entropy loss and an Adam optimizer with initial parameters of $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1 \times 10^{-7}$ (Kingma & Ba, 2014). Training data was batched with a batch size of 20. After each training epoch, model accuracy evaluated on the validation set for the fold was recorded. This validation accuracy was used for reducing the learning rate after plateauing: if the validation accuracy does not improve after 10 epochs, then the Adam learning rate is multiplied by 0.1 until a minimum learning rate of 1×10^{-8} is achieved (You, Long, Wang, & Jordan, 2019). After 120 epochs, the epoch with the highest validation accuracy is selected as the final trained model for the fold. Training times for traditional classifiers on the tomato dataset are shown in Table 4. Note that the number of parameters and depth of these models are largely unrelated to the number of classes, and training is done in number of epochs, so the total training time is linear in the amount of training data.

PCA models were trained by extracting the principal components from the training data and retaining only a certain number of them (referred to as the latent dimension). Through testing, it was found that this latent dimension could bias the EOCC toward classes with a higher latent dimension. In particular, it was found that discriminative performance suffered if different classes were represented by a different number of principal components or if the chosen principal components differed too much in the amount of information they captured. Therefore, the latent dimension was kept the same between classes. The validation set was used to perform an exhaustive search for the best latent dimension.

The AE models were trained on each class for 120 epochs with MAE loss and an Adam optimizer with initial parameters of $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1 \times 10^{-7}$ (Willmott & Matsuura, 2005). Training data was batched with a batch size of 20. Learning rate plateauing was used as described above.

A StyleGAN was trained on each class for 800 kimgs with $\gamma = 1$ and StyleGAN 2 loss, with an Adam optimizer with parameters of $\alpha = 0.001$, $\beta_1 = 0$, $\beta_2 = 0.99$, and $\varepsilon = 1 \times 10^{-8}$. Training data was batched with a batch size of 32, which did not fit in the 24 GB of VRAM available, so gradient accumulation was used with individual batches of size 8. Progressive growing is used to avoid mode collapse and segment "styles" (latent variables) among coarse and fine features - a key advantage of StyleGAN. After each StyleGAN has finished training, the final generator and discriminators from across training after every 20 kimgs were kept for use in models. Then, this generator was used to generate 1200 synthetic images, which were used together with the saved latent variables (styles) used to generate them to train an inversion network with MAE loss against the latent variables; an Adam optimizer with initial parameters of $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1 \times 10^{-7}$; and learning rate plateauing. In this way, the inversion network learns to "invert" synthetic images back to the latent variables used to produce them. This training step was continued for 20 epochs, after which an AE was constructed with the output from the inversion network fed into the inputs for the generator. This AE is trained on real training images from each fold for 100 epochs with MAE loss; an Adam optimizer with initial parameters of $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1 \times 10^{-7}$; and learning rate plateauing. After the inversion network was trained, then the validation set was used for model fine-tuning. An exhaustive search was used for jitter parameters — choosing $\mu = 0$ and $\sigma \in \{0.01, 0.1, 0.5, 1, 5\}$.

Training times for the AEs and StyleGANs on the Tomato dataset can be found in Table 5. Note that inversion network training time is in addition to the StyleGAN training time. Due to the way the StyleGAN is trained (in # of kimgs rather than epochs), the training time is invariant to the size of the dataset, but a separate StyleGAN must be trained for each class, so total training time is linear in the number of classes in the dataset. For the AE and inversion network, however, training is done in epochs, so training time is linear in the amount of training data in the dataset.

5.1.1. Modifying class structure

To compare the flexibility of EOCC to that of traditional classifiers, we examined how these models perform when some amount of the training and test sets have been slightly modified — such as by adding or removing a class from the set under consideration. How to train ensembles in such a scenario is very straightforward — simply remove any OCC from the ensemble that corresponds to any removed classes, or train any additional classifiers for additional classes and add them to the ensemble, then leave the rest of the classifiers untouched. For the class classifiers, however, there are many different approaches each with their own tradeoffs. The best option for performance would be to entirely retrain the model from scratch with the new set of classes, but this would involve using little to no work already done and will take just as long to retrain as it took to train in the first place. Another option is to freeze the feature extractor (where a bulk of the parameters of these models belong) and retrain just the classifying head of the model, which has shown to be an effective way of "transfer learning" from one dataset to another (Zhu et al., 2011).

To observe the differences in these approaches, every traditional classifier model was retrained after its initial training described above into an additional model. The "feature extraction" blocks of the models are frozen — not to be retrained, and the fully-connected classification head was reset with randomized weights and retrained using the new altered training and validation sets for the same number of epochs as the original models, then the best-performing model on the validation set was selected as the final, altered model.

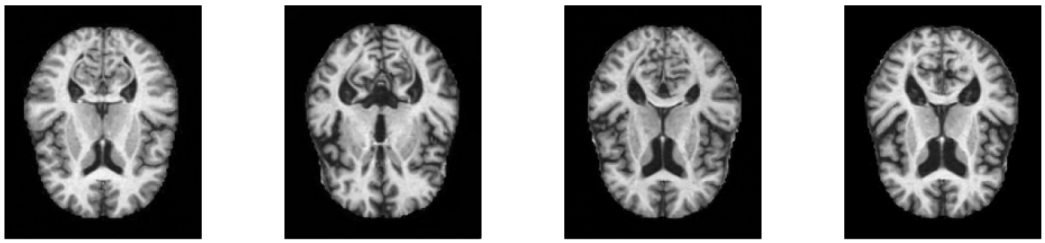


Fig. 11. Some examples of synthetic Alzheimer's images generated by StyleGAN for each class.

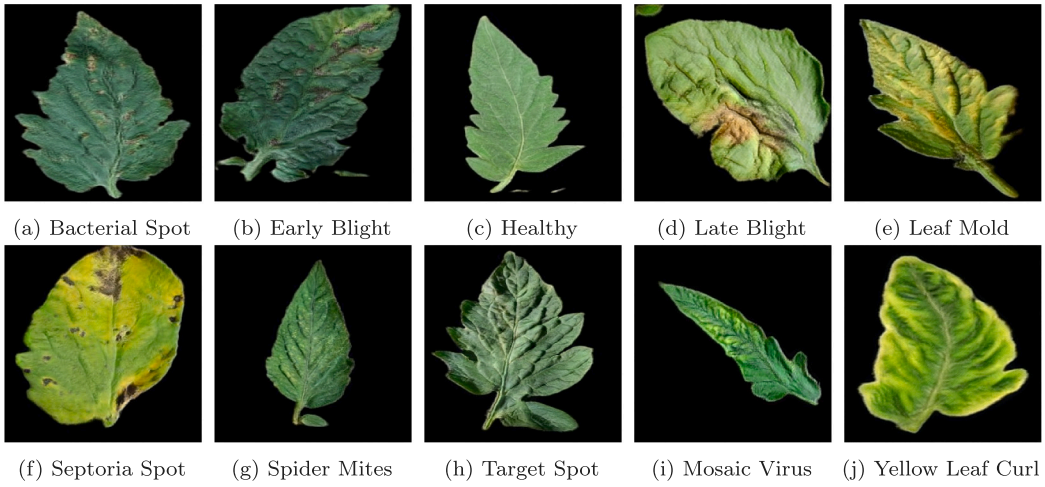


Fig. 12. Some examples of synthetic Tomato images generated by StyleGAN for each class.

5.1.2. Synthetic dataset augmentation

As an additional side effect to training GANs for each of the classes as a single class classifier, these GANs can also produce synthetic data that can be used to augment the original training data. It is well known that doing so can greatly increase model performance, especially in the cases of low or imbalanced data, as is common with medical data (Bowles et al., 2018; Frid-Adar, Klang, Amitai, Goldberger, & Greenspan, 2018; Wu, Wu, Cox, & Lotter, 2018). To take advantage of this, each PCA and AE classifier was trained with synthetic data generated by the trained StyleGANs. Synthetic data was incorporated into the training set for these classes at 4 different ratios of the original training set — 25%, 50%, 75%, and 100% to compare how the amount of synthetic data v.s. original data affects the performance of the resultant model depending on the dataset. Otherwise, the training of these augmented models is identical to what is described above. Examples of synthetic data can be seen in Figs. 11 and 12. It should be noted that since the way GANs are typically trained involves incorporating synthetic data generated by that very GAN (as "false" data that the discriminator must classify), using GAN-generated synthetic images to manually augment the training set of another GAN does not necessarily make sense, and would degrade the performance of GANs trained this way. Therefore, the only methods examined with synthetic dataset augmentation were PCA and AEs.

5.2. Testing

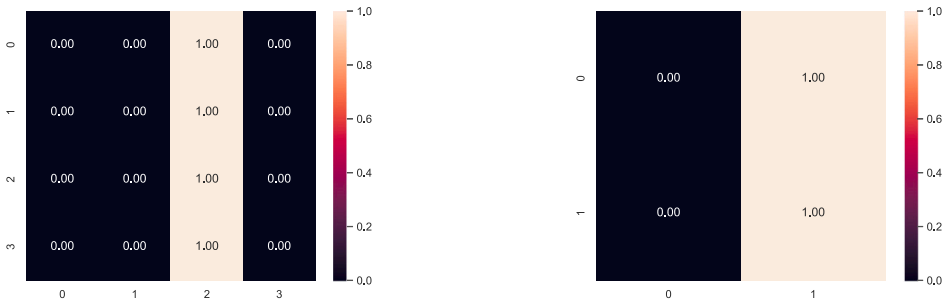
All models were evaluated using classification accuracy and are reported as the mean classification accuracy and standard deviation over all 5 training folds. For comparison, results for traditional classifiers can be found in Table 6 for all training sets. In general, EfficientNet B5 performs the best, in line with other prior works on the PlantVillage dataset, while VGG-19 and Resnet-50 fail to pick any class other than the most numerous on some of the sets (as evidenced by the 0.0000 standard deviation and can be seen in Fig. 13). Due to the way these models are trained, this is practically the minimum accuracy for these traditional classifiers.

Table 6
Multi-class classification performance using traditional deep classifiers on several sets of data. Mean test accuracy ± standard deviation across 5 folds. Best performing model for each set is highlighted in bold.

	VGG-19	Resnet-50	EfficientNet B3	EfficientNet B5
Alzheimer's	.5004 ± .0000	.5004 ± .0000	.5523 ± .0433	**.5820 ± .0216**
Corn	.8730 ± .0249	.4880 ± .0890	.9200 ± .0154	**.9300 ± .0195**
Strawberry	.7094 ± .0000	.7094 ± .0000	**.8487 ± .1831**	.5761 ± .3444
Potato	.4658 ± .0000	.5006 ± .0706	.9540 ± .0117	**.9547 ± .0160**
Apple	.8439 ± .0193	**.9911 ± .0072**	.9228 ± .0252	.9148 ± .0226
Tomato	.9180 ± .0107	.9711 ± .0032	.9759 ± .0034	**.9764 ± .0033**

Confusion matrices for test results with some select classifiers and datasets can be found in Fig. 14. The chosen datasets (Alzheimers and Potato) contain a class with very few samples compared to some other classes (class 1 in both cases - 1% and 6% of the total samples), and therefore the classifiers are biased toward not choosing those classes due to prior probabilities. This can be observed as the relatively dark second column in each of the confusion matrices.

Ensemble model results can be found in Table 7. StyleGAN-IJ performs best overall, sometimes not outperforming the normal inverted StyleGAN, which otherwise outperforms the other OCCs. StyleGAN-DH is not able to outperform the standard AE, while the StyleGAN-ID was able to outperform the standard AE, although never outperforming the standard StyleGAN-I classifiers. None of the ensembles were able to match the performance of either of the EfficientNet models in any dataset, although several models were able to beat VGG-19 and Resnet-50 performance in some of the classes. It is worth noting



(a) Alzheimers (b) Strawberry

Fig. 13. Confusion matrices of test results of Resnet-50 on datasets where the model could only choose the most common class.

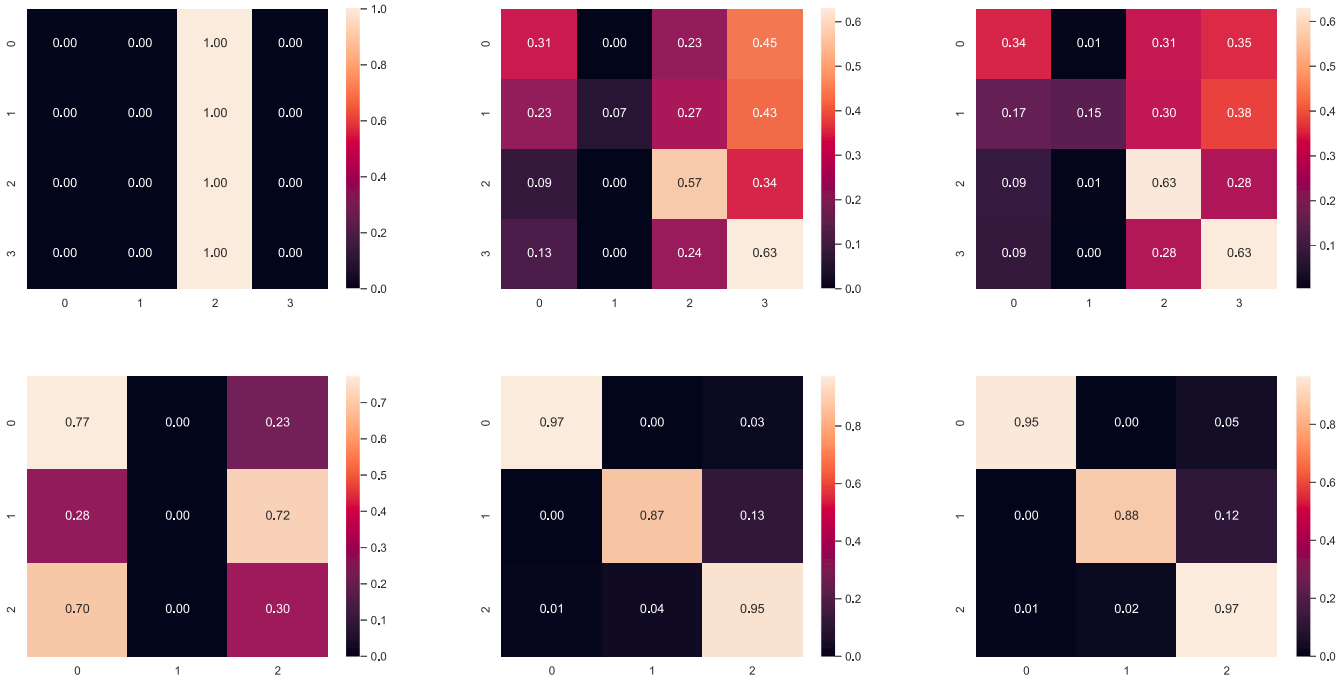


Fig. 14. Confusion matrices of test results of traditional classifiers on datasets containing classes with few training samples. The top row represents results from the Alzheimer's dataset, while the bottom is results from the Potato dataset. Classifiers are from left to right: Resnet-50, EfficientNet B3, and EfficientNet B5.

Table 7
Multi-class classification performance using ensembles of single-class classifiers. Mean test accuracy ± standard deviation across 5 folds. Best performing model for each set is highlighted in bold.

	PCA	AE	StyleGAN-I.	StyleGAN-IJ	SyleGAN-DH	StyleGAN-ID
Alzheimer's	.4746 ± .0415	.4973 ± .0375	**.5640 ± .0289**	.5480 ± .0317	.4558 ± .0421	.5152 ± .0317
Corn	.7565 ± .0368	.7270 ± .0271	**.8696 ± .0312**	.8409 ± .0304	.7704 ± .0217	.7948 ± .0275
Strawberry	.6154 ± .1712	.6966 ± .0943	.7350 ± .1287	**.7650 ± .0842**	.5726 ± .1735	.6752 ± .1158
Potato	.7671 ± .0315	.7547 ± .0214	.8230 ± .0301	**.8634 ± .0276**	.6801 ± .0223	.7578 ± .0251
Apple	.7523 ± .0381	.7684 ± .0318	.8346 ± .0271	**.8768 ± .0309**	.7270 ± .0395	.8127 ± .0294
Tomato	.7499 ± .0257	.7712 ± .0138	.8117 ± .0184	**.8488 ± .0179**	.7348 ± .0241	.8168 ± .0239

that some models perform worse than the minimum accuracy for the traditional classifiers mentioned above (this can be seen in the Alzheimer's dataset), since the way these ensembles are trained does not generally offer the ability to only pick a single class, regardless of how numerous that class is. Therefore, for very imbalanced data, it may be worth incorporating some measure of prior probability into the ensemble decision process to increase the effective minimum accuracy of such models.

Confusion matrices for test results with select ensembles and the same datasets as above can be found in Fig. 15. Note that since prior probabilities have less of an effect on these models as traditional classifiers — their minimum performance suffers, but they are not as biased as traditional classifiers when classifying classes with few samples. This can be seen from the lighter second columns in these confusion matrices.

5.2.1. Modifying class structure

Table 8 shows the change in model accuracy after some number of classes have been removed from the test set, and none of the models have been retrained. Instead, the output nodes corresponding to the removed classes in the traditional classifiers have simply been dropped before the last softplus activation — meaning if the classifier would have previously classified a given image as a removed class, it would instead choose the next most likely class. This means no additional

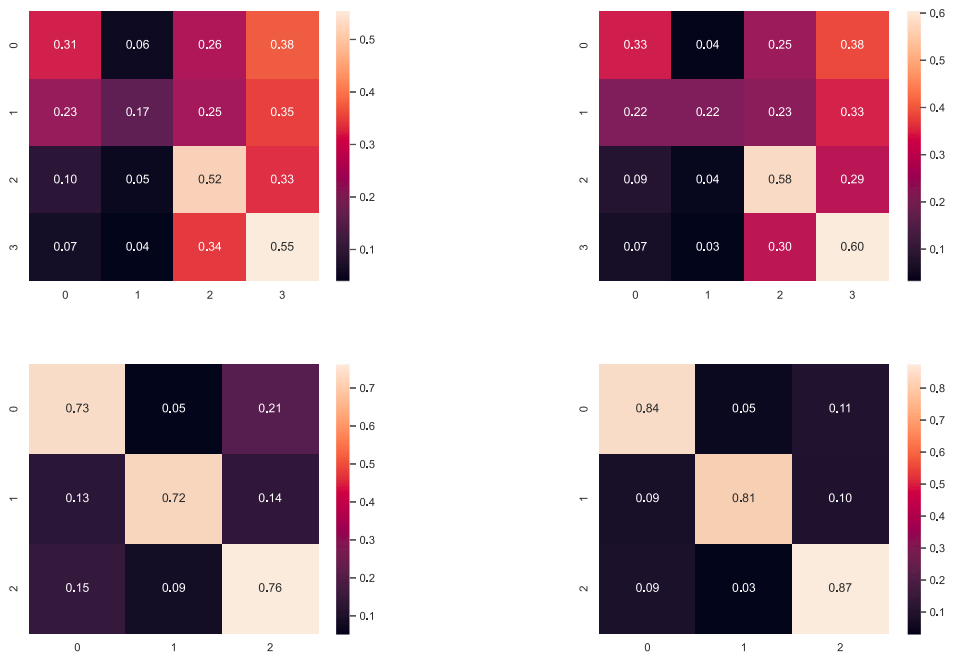


Fig. 15. Confusion matrices of test results of single-class ensembles on datasets containing classes with few training samples. The top row represents results from the Alzheimer's dataset, while the bottom is results from the Potato dataset. Classifiers are from left to right: AE and StyleGAN-IJ.

Table 8
Change in mean model performance over 5 folds when some number of classes have been removed from each set, and the traditional classifiers have not been retrained. In parentheses is the amount the removed classes contributed to the training set. Most improved model is highlighted in bold.

		Traditional				Ensembles		
	# classes (% data) removed	VGG-19	Resnet −50	Efficient-Net B3	Efficient-Net B5	PCA	AE	StyleGAN-IJ
Alzheimer's	1 (13.99%)	+.0814	+.0814	+.0657	+.0482	+.0738	**+.0970**	+.0837
Corn	1 (25.56%)	−.0561	+.0513	−.0529	−.0478	+.0412	+.0574	**+.0603**
Tomato	4 (40.75%)	−.0863	−.0951	−.0835	−.0717	+.0583	+.0690	**+.0725**

Table 9
Change in mean model performance over 5 folds when some number of classes have been removed from each set, and only the classifying heads of the traditional classifiers have been retrained. The most improved model is highlighted in bold.

		Traditional				Ensembles		
	# classes (% data) removed	VGG-19	Resnet −50	Efficient-Net B3	Efficient-Net B5	PCA	AE	StyleGAN-IJ
Alzheimer's	1 (13.99%)	+.0814	+.0814	+.0752	+.0614	+.0738	**+.0970**	+.0837
Corn	1 (25.56%)	−.0218	−.0172	+.0147	+.0219	+.0412	+.0574	**+.0603**
Tomato	4 (40.75%)	−.0106	−.0274	−.0302	−.0253	+.0583	+.0690	**+.0725**

time has been spent modifying any of the models after initial training. Due to results found in Table 7 above, a few representative models and datasets have been chosen to be tested. As can be seen, traditional classifiers generally suffer performance from taking this action, while the ensembles improve their performance. The improved performance from StyleGAN-IJ allows it to perform better than all other models in some classes, which demonstrates the flexibility of the ensemble methods.

Similar changes in model accuracy can be found in Table 9, where the traditional classifiers have had their classifying heads retrained, as detailed in Section 5.1.

5.2.2. Synthetic dataset augmentation

Improvements in performance to EOCC due to synthetic dataset augmentation can be found in Table 10. Note that no StyleGAN-based models are included, due to the way GANs are trained already including synthetic data. In general, including 75% of real training data as additional synthetic training data performs the best, except in the Alzheimer's dataset. This is likely due to the relative complexity of this dataset compared to the PlantVillage dataset and the lack of training data in some of the classes.

6. Conclusions and future work

This study addressed the problem of implementing MCC using EOCC. In this context, we introduced several new OCCs and demonstrated several benefits of EOCC besides classification accuracy. Traditional classifiers, including VGG-19, ResNet-50, and EfficientNet, were trained and compared with EOCC using PCA and AE OCCs, as described in previous research. Additionally, we introduced several GAN-based OCCs utilizing StyleGAN. Following the training of the StyleGAN generator, high-quality synthetic images of a class were generated. An inversion network was then trained to invert these images into their latent representations, which were subsequently used for image generation. This process enabled the training of deeper AEs, facilitating the development of more sophisticated networks. Using reconstruction

Table 10
Change in mean ensemble model performance over 5 folds when trained on different amounts of extra synthetic data generated by StyleGAN. Most improvement is highlighted in bold.

	PCA				AE			
	25%	50%	75%	100%	25%	50%	75%	100%
Alzheimer's	+.0111	+.**0156**	−.0083	−.0214	+.0119	+.**0375**	+.0139	−.0161
Corn	+.0292	+.0435	+.**0508**	+.0358	+.0399	+.0497	+.**0609**	+.0372
Strawberry	+.0282	+.0316	+.**0350**	+.0128	+.0265	+.0308	+.**0410**	+.0214
Potato	+.0143	+.0317	+.**0429**	+.0230	+.0335	+.0422	+.**0472**	+.0360
Apple	+.0283	+.0350	+.**0410**	+.0156	+.0376	+.0367	+.**0439**	+.0190
Tomato	+.0238	+.0258	+.0279	+.**0478**	+.0277	+.0320	+.0310	+.**0520**

error as a metric for class confidence in this way improves upon the accuracy in previous models. Furthermore, introducing Gaussian noise into the latent representations during the generation stage, where fine features are created and noise is resampled multiple times to form a consensus vote, further enhanced classification accuracy.

We demonstrated that the discriminator network of GANs could also serve as an OCC. In one approach, discriminators were saved at set intervals during training, and an image was evaluated using all of them. The standard deviation of the scores obtained provided a metric for class confidence. However, an EOCC employing this method did not outperform earlier approaches. In another approach, images were passed through the inversion network before being evaluated by the discriminator. The discriminator's output score served as a class confidence metric. While this EOCC variant outperformed previous methods, it did not surpass the performance achieved using inversion networks with reconstruction loss alone.

Our study also showcased the flexibility of EOCC compared to traditional classifiers, particularly in scenarios involving minimally altered class sets, such as adding or removing classes. Furthermore, we demonstrated that StyleGAN-based approaches could enhance the classification performance of other models through synthetic data augmentation. Finally, we improved the rigor of benchmarks on the PlantVillage dataset by introducing 5-fold cross-validation, ensuring more robust evaluation metrics. While many of the methods discussed in this study currently fall short of traditional classification techniques in terms of accuracy, the gap is steadily narrowing. With the development of more sophisticated approaches, it is becoming increasingly feasible to match or even surpass the performance of traditional classification methods, while also offering greater model flexibility. This progress opens several promising avenues for future research.

First, the potential of diffusion models to function as single-class classifiers, similar to the role GANs have played, warrants exploration (Ho, Jain, & Abbeel, 2020). Recent advancements in diffusion models have made them more robust, leading to their adoption in image generation and anomaly detection tasks (Wolleb, Bieder, Sandkühler, & Cattin, 2022). Incorporating diffusion models could significantly enhance the accuracy of the GAN-based models discussed in this study. Their inherent noise-adding mechanism, which is central to their architecture, aligns well with techniques previously employed to improve GAN classification performance. Beyond diffusion models, optimizing the latent space of OCCs is another promising direction. For example, combining AE with discriminator networks has been shown to produce more compact and informative latent spaces (Akcay, Atapour-Abarghouei, & Breckon, 2018; Sabokrou, Khalooei, Fathy, & Adeli, 2018). Exploring these methods further could yield significant improvements.

Second, alternative loss functions and reconstruction error metrics could be investigated. Metrics like MAE and RMSE tend to emphasize coarse image features over fine details, as larger pixel variations dominate the overall error calculation. However, fine-grained features often hold critical classification power, especially in many datasets. Using localized reconstruction error metrics to prioritize fine features could substantially enhance classification performance. Third, employing an MC-EEOCC approach with ensembles of homogeneous or heterogeneous OCCs could potentially achieve superior results. Lastly, methods for normalizing confidence scores across classes deserve attention. Without such normalization, classifiers may exhibit biases toward classes with inherently higher confidence scores, leading to skewed predictions. Addressing this issue could further improve the fairness and accuracy of classification models.

CRediT authorship contribution statement

Alexander Novotny: Contributed to the main ideas, Validation, Writing – original draft. **George Bebis:** Conceptualization, Supervision, Designed experiments, Writing – review & editing. **Alireza Tavakkoli:** Contributed to the main ideas, Supervision, Provided feedback . **Mircea Nicolescu:** Contributed to the main ideas, Supervision, Provided feedback.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Institute of Food and Agriculture/USDA, USA, Award No. 2020-67021-30754.

Data availability

Data will be made available on request.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. Software available from tensorflow.org.

Akcay, S., Atapour-Abarghouei, A., & Breckon, T. P. (2018). GANomaly: Semi-supervised anomaly detection via adversarial training. In *LNCS: Vol. 11363, Asian conference on computer vision*.

Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics*, *61*, Article 101182.

Baggenstoss, P. (2004). Class-specific classifier: Avoiding the curse of dimensionality. *IEEE Aerospace and Electronic Systems Magazine*, *19*(1), 37–52.

Ban, T., & Abe, S. (2006). Implementing multi-class classifiers by one-class classification methods. In *IEEE international joint conference on neural network proceedings* (pp. 327–332).

Blake, C., & Merz, C. (1998). *UCI repository of machine learning databases*. Irvine, CA (Online): Dept Inform Comput Sci, Univ California, http://kdd.ics.uci.edu/.

Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., et al. (2018). GAN augmentation: Augmenting training data using generative adversarial networks. arXiv:1810.10863.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. In *ACM SIGMOD international conference on management of data* (pp. 93–104).

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3).

Creswell, A., & Bharath, A. A. (2019). Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(7), 1967–1974.

Dietterich, T. (2000). Ensemble methods in machine learning. In *Multiple classifier systems* (pp. 1–15). Berlin, Heidelberg: Springer Berlin Heidelberg.

Dietterich, T., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Dubey, S. (2019). *Alzheimer's dataset (4 class of Images)*: *Technical report*, Kaggle, https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. Wiley-Interscience.

Duin, R. P. (2022). The combining classifier: to train or not to train? In *16th international conference on pattern recognition*.

Epstein, D., Park, T., Zhang, R., Shechtman, E., & Efros, A. A. (2022). Blobgan: Spatially disentangled scene representations. In *European conference on computer vision* (pp. 616–635). Springer.

Fernando, T., Gammulle, H., Denman, S., Sridharan, S., & Fookes, C. (2021). Deep learning for medical anomaly detection – A survey. *ACM Computing Surveys*, *54*(7).

Fragoso, R. C., Cavalcanti, G. D., Pinheiro, R. H., & Oliveira, L. S. (2021). Dynamic selection and combination of one-class classifiers for multi-class classification. *Knowledge-Based Systems*, *228*, Article 107290.

Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). Synthetic data augmentation using GAN for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging* ISBI 2018, (pp. 289–293).

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*(8), 1761–1776.

Garcia, K. D., de Sá, C. R., Poel, M., Carvalho, T., Mendes-Moreira, J., Cardoso, J. M., et al. (2021). An ensemble of autonomous auto-encoders for human activity recognition. *Neurocomputing*, *439*, 271–280.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press, http://www.deeplearningbook.org.

Hadjadji, B., Chibani, Y., & Guerbai, Y. (2017). Combining diverse one-class classifiers by means of dynamic weighted average for multi-class pattern classification. *Intelligent Data Analysis*, *21*(3), 515–535.

Hao, P., Chiang, J., & Lin, Y. (2009). A new maximal-margin spherical-structured multi-class support vector machine. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *30*, 98–111.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hearst, M., Dumais, S., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications*, *13*(4), 18–28.

Hempstalk, K., Frank, E., & Witten, I. H. (2008). One-class classification by combining density and class probability estimation. In *Lecture notes in computer science*: *Vol. 5211*, *Machine learning and knowledge discovery in databases*. ECML PKDD 2008.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, *33*, 6840–6851.

Hughes, D., Salathé, M., et al. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456). PMLR.

Juszczak, P., & Duin, R. P. W. (2004). Combining one-class classifiers to classify missing data. In F. Roli, J. Kittler, & T. Windeatt (Eds.), *Multiple classifier systems* (pp. 92–101). Springer Berlin Heidelberg.

Kang, S. (2022). Using binary classifiers for one-class classification. *Expert Systems with Applications*, *187*.

Kang, S., Cho, S., & Kang, P. (2015). Multi-class classification via heterogeneous ensemble of one-class classifiers. *Engineering Applications of Artificial Intelligence*, *43*, 35–43.

Kardan, N., & Stanley, K. O. (2018). Fitted learning: Models with awareness of their limits. arXiv:1609.02226 [cs.ai].

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4401–4410).

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8110–8119).

Khan, S., & Madden, M. (2014). One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, *29*(3), 345–374.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(3), 226–239.

Krawczyk, B., Galar, M., Woźniak, M., Bustince, H., & Herrera, F. (2018). Dynamic ensemble selection for multi-class classification with one-class classifiers. *Pattern Recognition*, *83*, 34–51.

Krawczyk, B., & Woźniak, M. (2014). Cytological image analysis with firefly nuclei detection and hybrid one-class classification decomposition. *Engineering Applications of Artificial Intelligence*, *31*, 126–135.

Krawczyk, B., Wozniak, M., & Cyganek, B. (2014). Clustering-based ensembles for one-class classification. *Information Sciences*, *264*, 182–195.

Krawczyk, B., Woźniak, M., & Herrera, F. (2015). On the usefulness of one-class classifier ensembles for decomposition of multi-classproblems. *PatternRecognition*, *48*, 3939–3982.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Lee, H., & Cho, S. (2006). The novelty detection approach for different degrees of class imbalance. In *Lecture notes in computer science*: *Vol. 4233*, *Neural information processing*. Springer,BerlinHeidelberg.

Lee, D., & Lee, J. (2007). Domain described support vector classifier for multi-classification problems. *Pattern Recognition*, *40*(1), 41–51.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *8th IEEE international conference on data mining* (pp. 413–422).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. *vol. 30*, In *Proc. ICML* (p. 3). Atlanta, GA.

Oza, P., & Patel, V. M. (2019). One-class convolutional neural network. *IEEE Signal Processing Letters*, *26*(2).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc..

Perera, P., Oza, P., & Patel, V. M. (2021). One-class classification: A survey. arXiv:2101.03064v1 [cs.cv].

Pimentel, M., Clifton, D., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–mICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18* (pp. 234–241). Springer.

Sabokrou, M., Khalooei, M., Fathy, M., & Adeli, E. (2018). Adversarially learned one-class classifier for novelty detection. In *IEEE/CVF conference on computer vision and pattern recognition*.

Salehi, M., Mirzaei, H., Hendrycks, D., Li, Y., Rohban, M. H., & Sabokrou, M. (2022). A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. arXiv:2110.14051v4 [cs.cv].

Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., & Boult, T. E. (2013). Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(7).

Scholkopf, B., Smola, A., & Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Scholkopf, B., Williamson, R., Smola, A., Taylor, J., & Platt, J. (1999). Support vector method for novelty detection. In *Neural information processing systems* (pp. 582–588).

Seliya, N., Zadeh, A. A., & Khoshgoftaar, T. M. (2021). A literature review on one-class classification and its potential applications in big data. *Journal of Big Data*, *8*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105–6114). PMLR.

Tax, D. M., & Duin, R. P. (2001). Combining one-class classifiers. In *2nd international workshop on multiple classifier systems* (pp. 299–308).

Tax, D. M., & Duin, R. P. (2004). Support vector data description. *Machine Learning*, *54*, 45–66.

Tax, D. M., & Duin, R. P. (2008). Growing a multi-class classifier with a reject option. *Pattern Recognition Letters*, *29*(10), 1565–1570.

Turk, M., & Pentland, A. (1991). Face recognition using eigenfaces. In *IEEE computer society conference on computer vision and pattern recognition* (pp. 586–591).

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, *30*(1), 79–82.

Wolleb, J., Bieder, F., Sandkühler, R., & Cattin, P. C. (2022). Diffusion models for medical anomaly detection. In *International conference on medical image computing and computer-assisted intervention* (pp. 35–45). Springer.

Wu, E., Wu, K., Cox, D., & Lotter, W. (2018). Conditional infilling GANs for data augmentation in mammogram classification. In *Image analysis for moving organ, breast, and thoracic images: third international workshop, RAMBO 2018, fourth international workshop, BIA 2018, and first international workshop, TIA 2018, held in conjunction with MICCAI 2018, Granada, Spain, September 16 and 20, 2018, proceedings 3* (pp. 98–106). Springer.

Xia, W., Zhang, Y., Yang, Y., Xue, J.-H., Zhou, B., & Yang, M.-H. (2022). GAN inversion: A survey. arXiv:2101.05278.

Yakkundi, A. (2023). Alzheimer's disease dataset. Mendeley Data, V1, http://dx.doi.org/10.17632/Ch87yswbz4.1.

You, K., Long, M., Wang, J., & Jordan, M. I. (2019). How does learning rate decay help modern neural networks? arXiv preprint arXiv:1908.01878.

Zhang, Y., Zhang, B., Coenen, F., Xiao, J., & Lu, W. (2014). One-class kernel subspace ensemble for medical image classification. *EURASIP Journal on Advances in Signal Processing*.

Zhu, Y., Chen, Y., Lu, Z., Pan, S., Xue, G.-R., Yu, Y., et al. (2011). Heterogeneous transfer learning for image classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, *25*(1), 1304–1309.

Zhu, J.-Y., Krähenbühl, P., Shechtman, E., & Efros, A. A. (2016). Generative visual manipulation on the natural image manifold. In *Computer vision–ECCV 2016: 14th European conference, amsterdam, the netherlands, October 11-14, 2016, proceedings, part v 14* (pp. 597–613). Springer.