University of Nevada, Reno

Investigating Ensembles of Single-class Classifiers for Multi-class Classification

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Computer Science and Engineering

by
Alexander Novotny
Dr. George Bebis/Thesis Advisor

August 2023

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Alexander Novotny

entitled

Investigating Ensembles of Single-class Classifiers for Multi-class Classification

be accepted in partial fulfillment of the
requirements for the degree of

Master of Science

George Bebis, Ph.D.
Advisor

Alireza Tavakkoli, Ph.D.
Committee Member

Eric Olson, Ph.D.
Graduate School Representative

Markus Kemmelmeier, Ph.D., Dean
Graduate School

August, 2023

Abstract

Traditional methods of multi-class classification in machine learning involve the use of a monolithic feature extractor and classifier head trained on data from all of the classes at once. These architectures (especially the classifier head) are dependent on the number and types of classes, and are therefore rigid against changes to the class set. For best performance, one must retrain networks with these architectures from scratch, incurring a large cost in training time. As well, these networks can be biased towards classes with a large imbalance in training data compared to other classes. Instead, ensembles of so-called "single-class" classifiers can be used for multi-class classification by training an individual network for each class.

We show that these ensembles of single-class classifiers are more flexible to changes to the class set than traditional models, and can be quickly retrained to consider small changes to the class set, such as by adding, removing, splitting, or fusing classes. As well, we show that these ensembles are less biased towards classes with large imbalances in their training data than traditional models. We also introduce a new, more powerful single-class classification architecture. These models are trained and tested on a plant disease dataset with high variance in the number of classes and amount of data in each class, as well as on an Alzheimer's dataset with low amounts of data and a large imbalance in data between classes.

Contents

List of Tables

List of Figures

1 Introduction

Multi-class classification is the supervised learning problem of separating samples of data into one of more than two classes (as opposed to binary classification). Many real-world applications rely on multi-class classification techniques, such as facial recognition, optical character recognition for reading handwriting, and disease detection. Traditionally, approaches to multiclass-classification involve using a single monolithic model which is responsible for extracting features from the original data representation and classifying those features for every single class. However, this approach has a flaw - if we want to make changes to the class structure of the dataset, such as by adding or removing classes, or splitting and fusing them, we must retrain our model (or some part of it) from scratch. As an alternative to this approach, we can instead use ensembles of single-class classifiers. Single-class classification, also known as intruder detection, anomaly detection, novelty detection, and concept learning, is the *unsupervised* learning problem of determining if a sample of data belongs to a single class, when that class is the only class that has been seen before [32, 17]. A single-class classifier trained on only a single class of data determines a score called a "likeness score" or "intruder score" that determines how close a sample is to being of that class. This likeness score can then be thresholded for purposes of single-class classification - if the resultant score is above or below a certain threshold, then the sample belongs to that class. Otherwise, it does not. An ensemble of these classifiers can be used for multi-class classification by producing a likeness score for each class under consideration, and then classifying as the class with the highest likeness score. Such ensembles have been used for several applications [16, 2, 32, 23, 10, 4, 9]. However, prior studies on these ensembles have failed to compare them to traditional neural classifiers. As well, the latest approaches to these ensembles stop at using autoencoder-based single-class classifiers. Autoencoders suffer from training stability problems if made too deep, and are therefore difficult to get sophisticated enough for computer vision tasks. This study benchmarks existing autoencoder-based ensembles against traditional neural classifiers and introduces several new techniques for neural network-based single class classification, some of which improve upon existing autoencoder-based ensembles.

This thesis is structured as follows: Section 2 details previous work on ensembles of single-class classifiers for multi-class classification. Section 3 presents an introduction and description of all new examined models, as well as a description on how to implement an ensemble of single-class classifiers for multi-class classification. Section 4 gives an overview on the types of data used for testing the models presented. Section 5 describes the process to obtaining the results given, presents these

results, and discusses them. Finally, Section 6 gives final conclusions and ideas for future work on improving ensembles of single-class classifiers for multi-class classification.

2 Background

The first example of a single-class classifier was given by Turk and Pentland [33], who used Principal Component Analysis (PCA) to determine if an image contained a face using reconstruction error (called "distance from face space"). As the original method for single-class classification, it had some problems that other models improve on; namely: the method is inherently linear, which means it fails to capture nonlinear relationships. To overcome this, Tax and Duin [31] introduced an SVM-based single-class classifier called Support Vector Domain Description (SVDD), which uses a Support Vector Machine (SVM) to introduce nonlinearity, similar to how they were used in traditional classifiers [12]. Rather than using an SVM to find the decision boundary between two classes using hyperplanes in a larger dimension, they used an SVM to find the minimal hypersphere containing all objects in a larger dimension. Then, the distance of a test object from the center of this sphere can be used as an intruder score, while the radius of the sphere is a natural threshold. Like classifying SVMs, this method has a lot of customizability through the choice of the kernel function used, and therefore has many hyperparameters to optimize, leading to difficulty in coming up with a well-performing model. Lee and Lee [23] used this method of single-class classification in an ensemble to solve the issue of how to use SVMs (only used for binary classification) in multi-class classification. Originally, one-vs-all and pairwise strategies were tried, but they were imperfect and left some regions of the data space unclassified or multiply classified. Instead, they used an ensemble of SVDDs, each trained on a class, which produce an intruder score for each class when fed a test object. They then take a step to convert these intruder scores into a pseudo-posterior probability, and choose the class with the highest probability. This probability conversion step does some amount of work to regularize the scores from the different classes, as the radius of each hypersphere is likely to be different (and therefore some classes are expected to have larger scores without being intruders). This method was able to outperform the binary SVMs and the (at the time) rudimentary neural network approach. Hao, Chiang, and Lin [10] went a step further and improved these results to incorporating the maximal margin technique used by classification SVMs and found hyperspheres for each class with maximal margins between the rest of the classes. Ban and Abe [2] introduce kernel principle component analysis as an improvement to SVDD when used as a single-class classifier, which combines PCA with

the kernel trick used to introduce nonlinearity in SVMs. This method performs better than SVDD on homogeneous datasets due to the inherent whitening process in PCA which rescales variance to be equal in all directions. Kang, Cho, and Kang [17] improves on these results by altering the ensemble from homogeneous to heterogeneous - by including multiple types of single-class classifiers per class, and combing the results for the final classification. Their results also include autoencoders, which often performed better than SVDD and PCA techniques. While the change from homogeneous to heterogeneous ensemble allows for a much more robust model, it also means that there is a final training process after all individual classifiers are finished training. This means that if there were a small change to the class set under consideration, the model would not be as quick to retrain, as this final training process would need to be repeated. As well, they only compare results of other single-class classifiers, rather than comparing to traditional multiclass classification techniques, such as the deep neural networks which were just starting to become popular. Garcia et al. [9] used an ensemble of autoencoders for human activity recognition. They were able to show improvements over other state of the art neural approaches, however, their dataset is much simpler than images used for computer vision, and the architecture used for the autoencoders used in the ensemble is fairly rudimentary.

In this study, we apply ensembles of single-class classification for multi-class classification of more complex computer vision datasets, introduce more deep-learning-based single-class classification approaches, and benchmark against standard deep learning classifiers for computer vision tasks.

3 Methodology

All single-class classifiers accept a single sample (in this case, an image) as input, and produce either a measure of how similar it is to the rest of the class the model was trained on (a "likeness score") or a measure of hose different is it (an "intruder score"), as is common with reconstruction-based approaches. To construct an ensemble of such classifiers, we simply take an input sample, pass it to all of the single-class classifiers, record the score, and use a simple meta-classifier to determine the class from these scores. An overview of this architecture can be found in fig. 1. In this study, we use the simple meta-classifier of picking 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 error associated with them than other classes. Then,

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 the validation set, however we were not able to find a method which consistently produced better results than simply picking lowest/highest score. There are some more sophisticated methods of accomplishing this, such as the heterogeneous approach proposed by Kang, Cho, and Kang [17], however we wished to avoid having to train the meta-classifier.

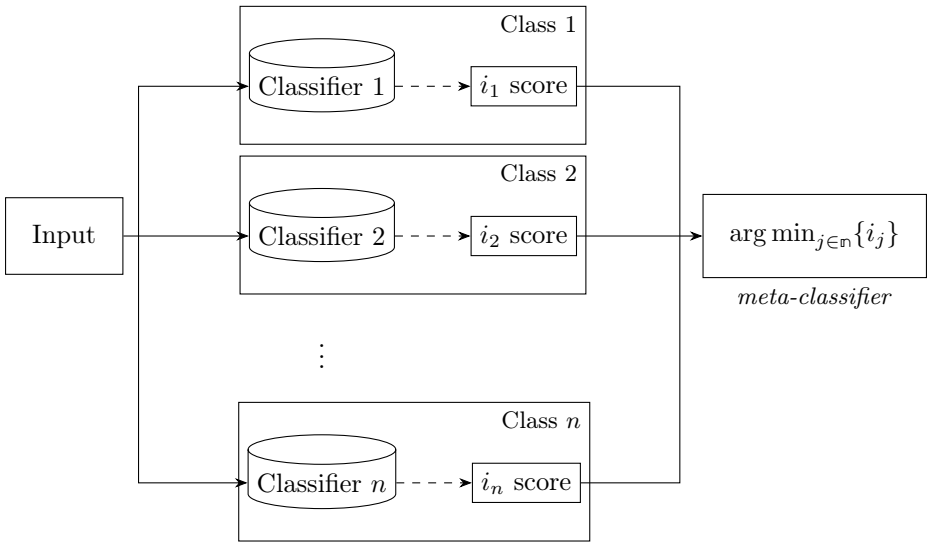


Figure 1: An overview of the proposed ensemble method architecture for classifying n classes.

3.1 Principle Component Analysis

PCA is used as a benchmark to compare against all of the other single-class classifiers, a technique first introduced by Turk and Pentland [33] in the context of face detection/recognition. All images in a class are converted to feature vectors, and a covariance matrix is computed. The eigenvectors of this matrix (the principal components) are sorted by decreasing eigenvalues, and only the first several are kept - how many determines the latent dimension. Each of these principle components is associated with a certain amount of information captured by each component - represented by the corresponding eigenvalue. Therefore, keeping the first several principal components retains the largest amount of information of the dataset, and therefore reconstructive power. Future images are projected onto these principal components and reconstructed, which can be used to calculate a reconstruction error by taking the magnitude of the difference between the original image and the reconstructed image (see fig. 2). The further an image is from the span of these principal components

(referred to as "distance from face space" in the original paper), the larger the reconstruction error. Therefore, this reconstruction error is a good choice of intruder score for single-class classification. The choice of principal components to keep during the reconstruction process for ensembles classifiers is fairly arbitrary - as there isn't necessarily a direct correlation between overall reconstructive power and discriminator power, since we cannot determine which principal components encode information common among other classes without training on those classes. By testing, however, it was found that discriminator performance suffered if different classes had a different number of principle components (the latent dimension) or the chosen principle components differed too much in captured information. Otherwise, a validation set could be used for finding the best set of principle components to keep using some type of search algorithm, such as exhaustive search or a genetic algorithm.

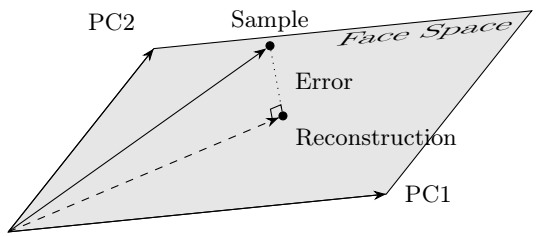


Figure 2: An illustration of the "face space" constructed by PCA, and where the reconstruction error is obtained from. Principal Components labeled as "PC#".

3.2 Autoencoder

The encoder half of the autoencoder architecture is made up of 4 downsampling blocks. Each downsampling 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 [25, 15]. The first downsampling block has 16 convolution filters, and each block after double the number of filters, therefore reducing the number of features after each block by half. Convolutional stride was chosen over pooling by test results. After the downsampling 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 downsampling blocks and the latent dimension.

The decoder half of the autoencoder 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 downsampling blocks, which are followed by 4 upsampling blocks. Each upsampling 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 [24]. The first upsampling block has as many convolutional filters as the last downsampling blocks, and each upsampling 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 upsampling blocks, and a final convolutional layer with no stride, no activation function, and 3 filters to produce the final resulting image.

A visual representation of the combined encoder and decoder architectures can be found in fig. 3.

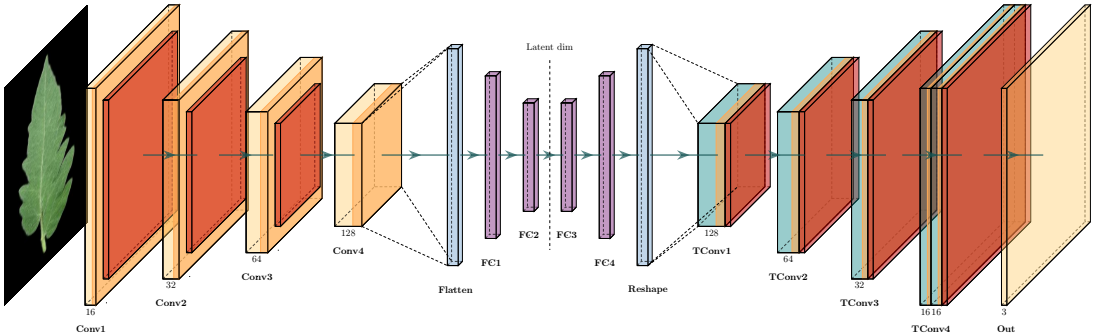


Figure 3: A visual representation of the proposed autoencoder architecture.

An image can be fed into the encoder half of the autoencoder, 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 autoencoder, 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, similarly to PCA, and can be used as an intruder score for single-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. As well, 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 autoencoders, there is no way to specify such an ordering. Moreover, the loss functions used to train autoencoders typically employ a global reconstruction error (such as in Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE)), which encourage 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 thing without having access to other classes' training data.

Attempts to increase the depth of the autoencoder to improve performance, either by increasing the number of up/down sampling blocks or by including more convolutional layer per block as in U-Net, were met with difficulties in training the model [28]. 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) doesn't necessarily fit our use case - the variables "skipping" across the bottleneck of the autoencoder are actually part of the latent dimension, and therefore we must either greatly enlarge the latent dimension (which is fine in the original use case of U-Net - image to image translation, rather than encoding) or reduce the capacity of each skip connection.

3.3 StyleGAN

To attempt to fix the shortcomings of a deep autoencoder, we introduce a Generative Adversarial Network (GAN)-based approach. GANs have several techniques that make them easier to train at significant depth, such as progressive growing [20]. As well, training a GAN allows you to train just the decoder part of the autoencoder first (as the generator of the GAN), and then train the encoder separately as an inversion network. StyleGAN uses skip connections from the latent variables to each upsampling block, rather than the skip connections that go across the bottleneck like U-Net [18]. 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. 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 [19].

To turn the generator of a GAN into a single-class classifier, an inversion network is needed [40, 5, 38]. The inversion network makes up the encoder half of the autoencoder, and learns to map images to the latent variables used to generate them (or as close an approximation as is possible). We use the feature extractor part of the StyleGAN classifier, as suggested by Epstein et al. [7]. The

feature extractor consists of several discriminator blocks, which are composed of two 3×3 convolution layers followed by a $2\times$ bilinear downsampling layer. A residual skip connection is added from the beginning of the block to the end, which consists of a single downsampling operation and a 1×1 convolution. Like the encoder described in the autoencoder architecture in section 3.2, the resolution is halved 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.

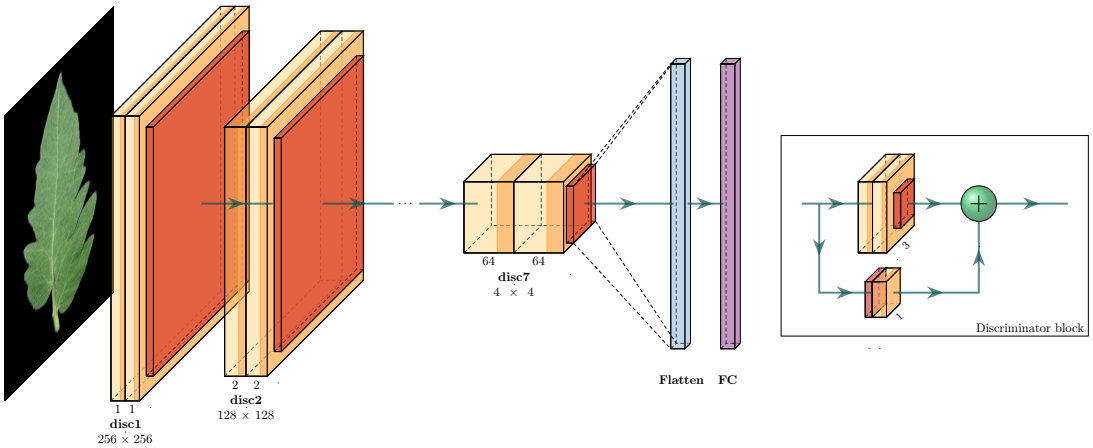


Figure 4: A visual representation of the proposed inversion network.

To improve classification results using the inverted StyleGAN, a "jitter" technique is used to emphasize differences in fine features between classes. Inspired by the sampling process of Variational Autoencoders, we introduce some Gaussian noise to the latent variables produced by the inversion network before reconstructing with the StyleGAN generator [22]. This Gaussian noise is only introduced to the latent variables at certain resolutions in the upsampling 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 upsampling blocks, where fine features are constructed. We sample the noise and reconstruct 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.

3.4 GAN Discriminator

While training a GAN, two networks are trained - the generator and the discriminator. A fair amount of work is put into training both networks, but after training, the discriminator is typically discarded and only the generator is used, as in the inversion network described above. To get more worth out of this training process, we have investigated the possibility of using a GAN discriminator as a single-class classifier. In that way, one could pick between the inversion techniques described above, discriminator techniques, or some combination of both after training a GAN.

The first technique we investigated was using discriminators directly - simply feed images to the discriminator, record the output scores, and threshold. Since the discriminator network has been trained so well to tell the difference between real images of a class and fake images, it should be able to tell that an image from another class isn't a real image of the class it was trained on. We first tried with the discriminator obtained at the end of training, but as can be seen in fig. 5, a reasonable discriminator is able to 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 that may not necessarily be even visible to the human eye, which won't be present in images from other classes. Therefore, real images from other classes have similar scores to real images from the trained class. This can be observed in fig. 6.

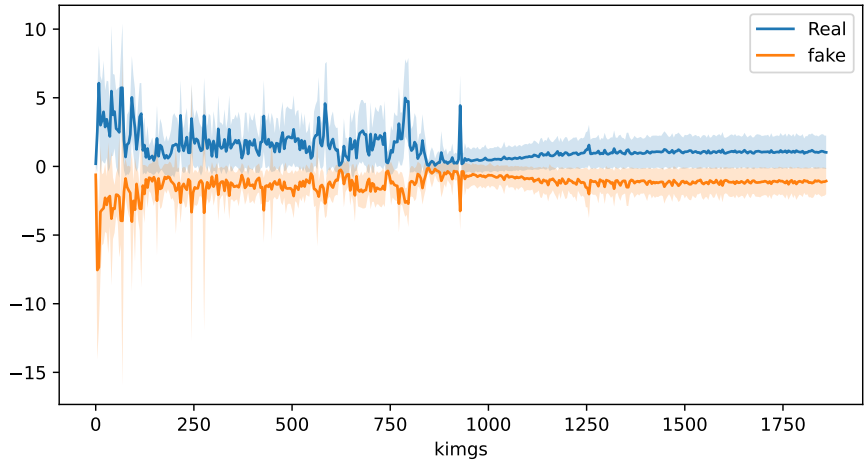


Figure 5: Tomato healthy class StyleGAN discriminator scores of real and fake images over training time. Scores captured as a moving average, with standard deviation shown in shaded regions.

To avoid this issue, discriminators from across the training process are examined. If an image from another class is evaluated by discriminators from across the training process, its scores will

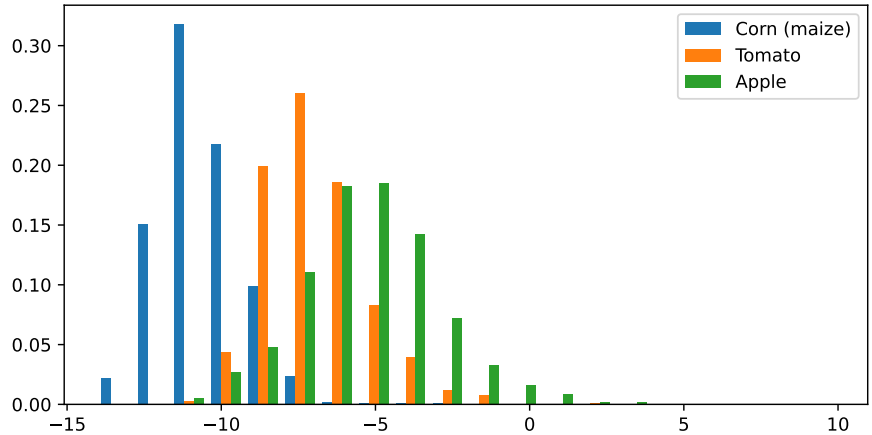


Figure 6: Histogram of discriminator scores of images from a variety of classes. The discriminator was trained on the Tomato class.

vary wildly until becoming more consistent as the training process goes on, as can be seen in fig. 7. In this way, we can take an image we want to classify, evaluate the discriminator score across all discriminators saved from training, and calculate the variance of these scores. Then, the variance can be used as an intruder score like reconstruction error. This method of single-class classification is referred to as "discriminator history".

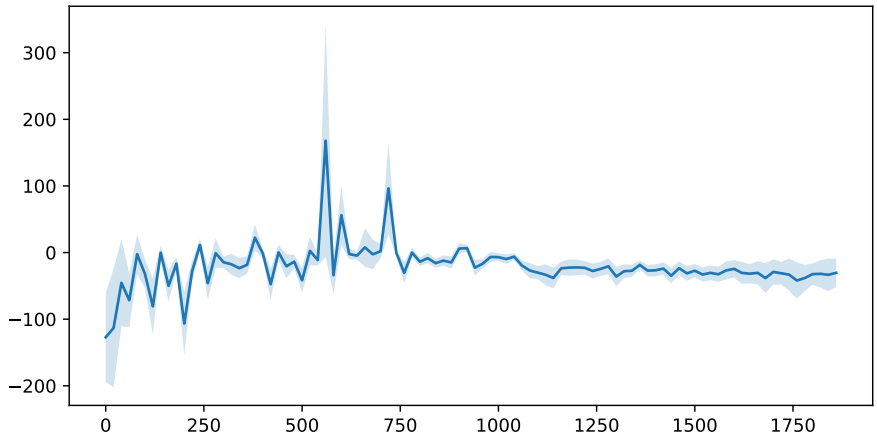


Figure 7: Mean discriminator scores of corn images evaluated on all discriminators trained on tomato images. Standard deviation shown in shaded regions.

Another way to avoid this issue is to ensure that all images seen by the discriminator are fake. We can use the inversion network described in section 3.3 to reconstruct all images before showing

them to the discriminator - that way all images have the defects introduced by the generator that the discriminator has learned, and the only difference in scores will be from the difference in classes. Then, the discriminator score can be used as a likeness score like reconstruction error - although inverted (the ensemble chooses the highest score). This method of single-class classification is referred to as "GAN with Discriminator".

4 Datasets

An overview of the different datasets examined in this study is available in table 1.

Table 1: Properties of the datasets used for training and testing. Images / Class given in mean ± std. deviation.

	Size (px)	Classes	Images / Class
Alzheimer's	176×208	4	1600 ± 1393
Corn	255×255	4	963 ± 313
Strawberry		2	783 ± 461
Potato		3	717 ± 489
Apple		4	793 ± 591
Tomato		10	1815 ± 1349

4.1 Plant Village

The Plant Village dataset was selected for this study due to its hierarchical nature [14]. The dataset is made up of images of plant leaves from 14 different plants (only 5 of which are examined in this study), each with multiple different diseases, for a total of 38 different classes. The dataset 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. Examples of the different plants examined in this study can be seen in fig. 8, and examples of the different diseases for the Tomato plant can be seen in fig. 9. 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 a perfect example of a dataset for which someone may want to change the class set under examination - such as by adding or removing relevant plants and diseases as the seasons change, or when there is a new outbreak. As well, some prior benchmarks for this dataset lack rigorous results by way of cross-validation, which provides us an opportunity to improve

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

Figure 8: Examples of some of the different types of plants included in PlantVillage.

these benchmarks [1]. Note 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.

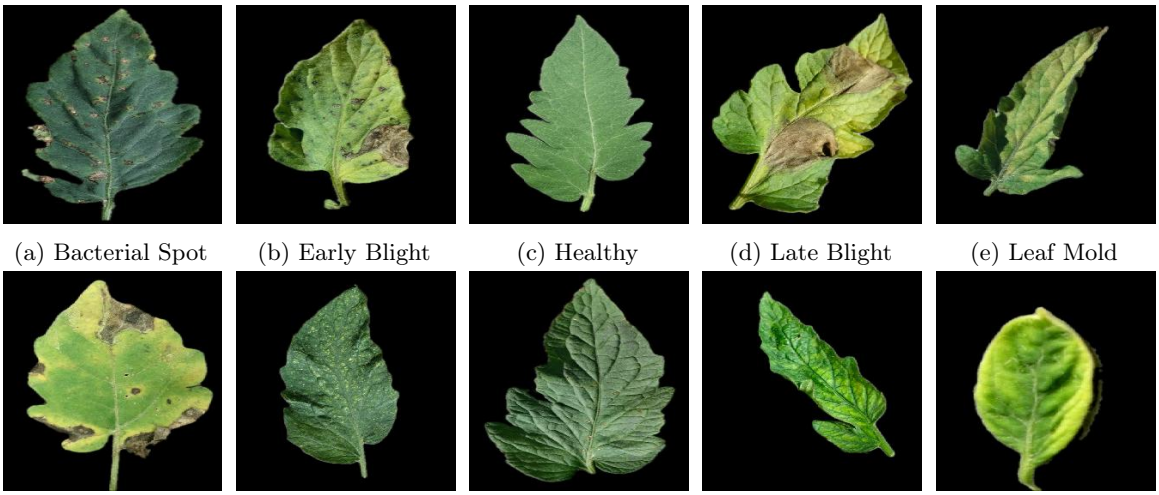
A list of all of the classes examined in this study and their sizes can be found in table 2. Note that the Potato and Tomato contain the largest outliers in terms of sample size with the Healthy and Yellow Leaf classes, respectively. Since these datasets contain such outliers, we will focus on them when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

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

Dataset	Classes & No. Samples				
Corn	Cercospora 513	Common Rust 1,192	Healthy 1,162	Northern Leaf Blight 985	
Strawberry	Healthy 456	Leaf Scorch 1,109			
Potato	Early Blight 1,000	Healthy 152	Late Blight 1,000		
Apple	Apple Scab 630	Black Rot 621	Cedar Apple Rust 275	Healthy 1,645	
Tomato	Bacterial Spot 2,127	Early Blight 1,000	Healthy 1,591	Late Blight 1,909	Leaf Mold 952
	Septoria Spot 1,771	Spider Mites 1,676	Target Spot 1,404	Mosaic Virus 373	Yellow Leaf Curl 5,357

4.2 Alzheimer's

The Alzheimer's dataset was selected for this study to observe performance of the proposed models on complex medical images with very little training data [6]. The dataset consists of single monochrome slices of a brain scan in patients with or without Alzheimer's. Examples of images from the different



(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

Figure 9: Samples from each of the different classes in the Tomato subset of PlantVillage.

classes of this dataset can be seen in fig. 10, and a breakdown of sample size by class can be found in table 3. All images are resized to 256×256 using 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.

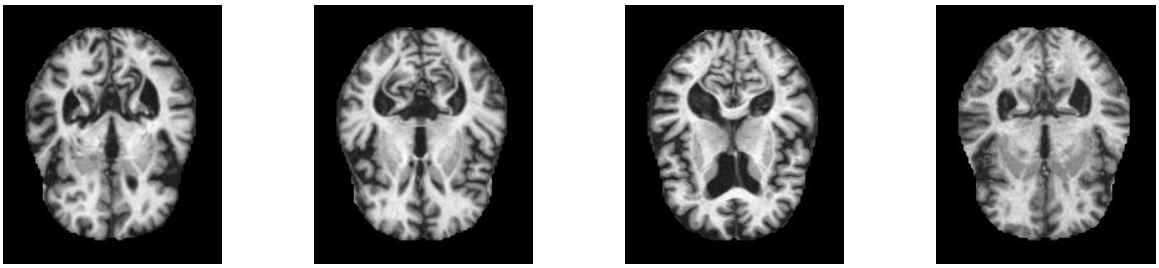
Table 3: Class breakdown of the Alzheimer's dataset by sample size.

Class	No. Samples
Healthy	2,560
Moderate	52
Mild	717
Very Mild	1,792

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, kept separate. Then the remaining data was split into 5 equal 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



(a) Healthy (b) Moderate (c) Mild (d) Very Mild

Figure 10: Samples from each of the different classes in the Alzheimer's dataset.

training. The StyleGAN models were developed and trained using the Pytorch Python framework due to the original work being done there, but the rest of the models were trained using the Tensorflow Python framework [27, 26]. 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 as comparison models. These architectures were VGG-19, Resnet-50, and EfficientNet, which are all well-known architectures in the field of multi-class classification [29, 11, 30]. As well, EfficientNet has some prior work in the PlantVillage dataset [1]. 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 $\varepsilon = 1 \times 10^{-7}$ [21]. 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 doesn't 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 [39]. 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 can be found 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.

Table 4: Mean total training time of traditional classifiers on the tomato subset over all 5 folds, in hours.

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

PCA models were trained by extracting the principal components of 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 ensemble classifier toward classes with a higher latent dimension, so the latent dimension was kept constant between classes. The validation set was used to perform an exhaustive search for the best latent dimension, and the best principal components to select to form the model, between 5 and 16 components.

The autoencoder 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 $\varepsilon = 1 \times 10^{-7}$ [35]. 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 didn't 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 are 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 autoencoder is constructed with the output from the inversion network fed into the inputs for the generator. This autoencoder 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 autoencoders 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 autoencoder and inversion network, however, training is done in epochs, so training time is linear in the amount of training data in the dataset.

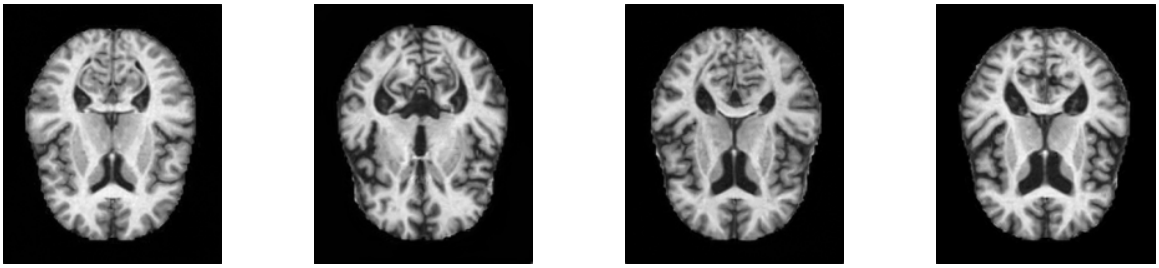
Table 5: Mean total training time of single-class classifiers on the tomato subset in hours.

Autoencoder	StyleGAN	Inversion Network
2.25	8.79	2.39

5.1.1 Modifying class structure

To compare the flexibility of ensembles of single-class classifiers to that of traditional classifiers, we examine 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 single-class classifiers from the ensemble that correspond 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 [41, 34].

To observe the differences in these approaches, every traditional classifier model is 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 is 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.



(a) Healthy (b) Moderate (c) Mild (d) Very Mild

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

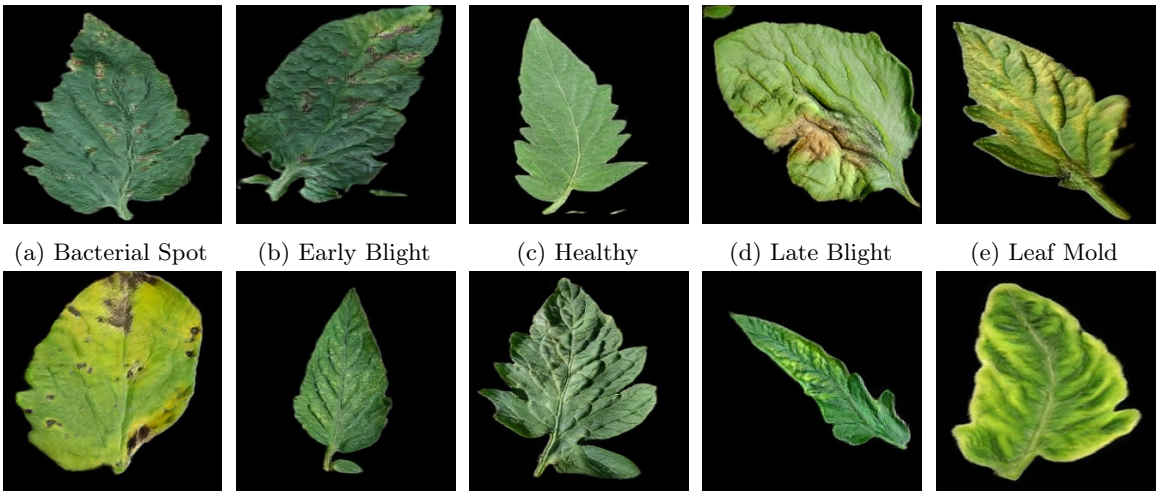
5.1.2 Synthetic dataset augmentation

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 [8, 3, 37]. To take advantage of this, each PCA and autoencoder classifier is trained with synthetic data generated by the trained StyleGANs. Synthetic data is 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 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.

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 doesn't necessarily make sense, and would degrade the performance of GANs trained this way. Therefore, the only methods examined with synthetic dataset augmentation are PCA and autoencoders.

5.2 Testing

All models are 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 .0000 standard



(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

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

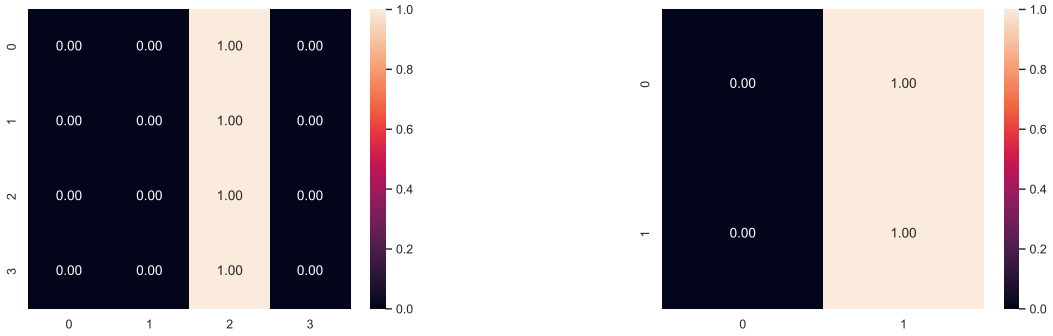
deviation and can been seen in fig. 13). Due to the way these models are trained, this is practically the minimum accuracy for these traditional classifiers.

Table 6: Multiclass 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 highlighted in bold.

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

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 towards 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. The inverted StyleGAN with the jitter technique performs best overall, sometimes not outperforming the normal inverted StyleGAN, which otherwise outperforms the other single-class classifiers. The discriminator history method isn't able to outperform the standard autoencoder, while the Inverted StyleGAN with the Discriminator is



(a) Alzheimers (b) Strawberry

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

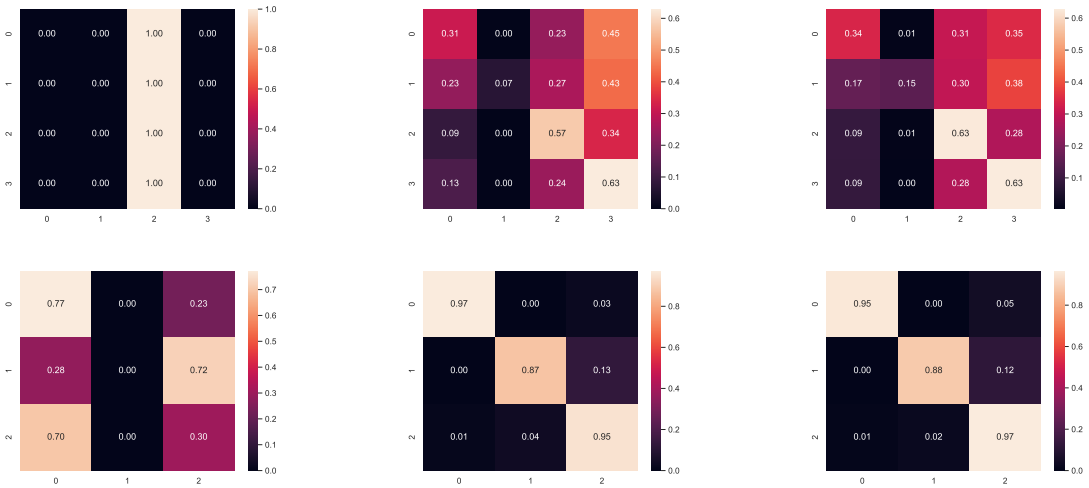


Figure 14: Confusion matrices of test results of traditional classifiers on datasets containing classes with few training samples. The top row represents results from the Alzheimer's dataset, while the bottom is results from the Potato dataset. Classifiers are from left to right: Resnet-50, EfficientNet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

able to outperform the standard autoencoder, although never outperforming the standard inverted StyleGAN classifiers. None of the ensembles are able to match the performance of either of the EfficientNet models in any dataset, although several models are able to beat VGG-19 and Resnet-50 performance in some of the classes. It is worth noting 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 doesn't 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.

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

	PCA	Autoencoder	Inv. StyleGAN	GAN w/ Jitter	Discr. History	GAN w/ Discr.
Alzheimer's	$.4746 \pm .0415$	$.4973 \pm .0375$	$\mathbf{.5640 \pm .0289}$	$.5480 \pm .0317$	$.4558 \pm .0421$	$.5152 \pm .0317$
Corn	$.7565 \pm .0368$	$.7270 \pm .0271$	$\mathbf{.8696 \pm .0312}$	$.8409 \pm .0304$	$.7704 \pm .0217$	$.7948 \pm .0275$
Strawberry	$.6154 \pm .1712$	$.6966 \pm .0943$	$.7350 \pm .1287$	$\mathbf{.7650 \pm .0842}$	$.5726 \pm .1735$	$.6752 \pm .1158$
Potato	$.7671 \pm .0315$	$.7547 \pm .0214$	$.8230 \pm .0301$	$\mathbf{.8634 \pm .0276}$	$.6801 \pm .0223$	$.7578 \pm .0251$
Apple	$.7523 \pm .0381$	$.7684 \pm .0318$	$.8346 \pm .0271$	$\mathbf{.8768 \pm .0309}$	$.7270 \pm .0395$	$.8127 \pm .0294$
Tomato	$.7499 \pm .0257$	$.7712 \pm .0138$	$.8117 \pm .0184$	$\mathbf{.8488 \pm .0179}$	$.7348 \pm .0241$	$.8168 \pm .0239$

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 conusion 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 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 the Inverted StyleGAN with the jitter technique

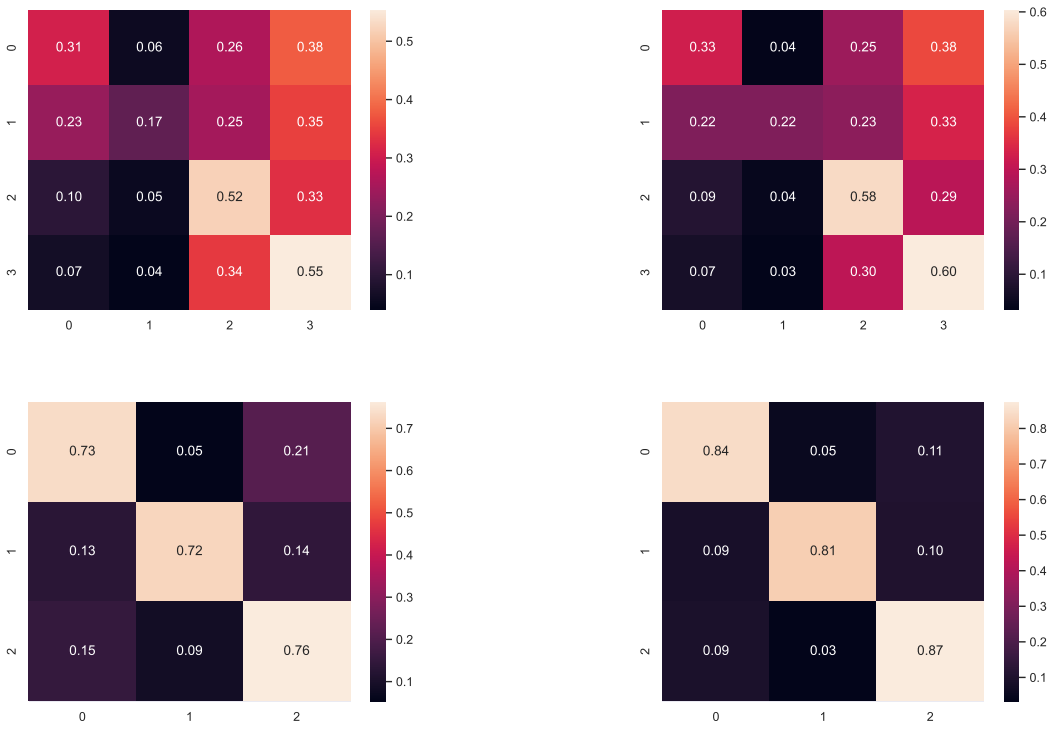


Figure 15: Confusion matrices of test results of single-class ensembles on datasets containing classes with few training samples. The top row represents results from the Alzheimer's dataset, while the bottom is results from the Potato dataset. Classifiers are from left to right: Autoencoder and Inverted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

allows it to perform better than all other models in some classes, which demonstrates the flexibility of the ensemble methods.

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 highlighted in bold.

		Traditional				Ensembles		
	# classes (% data) removed	VGG-19	Resnet -50	Efficient-Net B3	Efficient-Net B5	PCA	Auto-encoder	Inv. GAN w/ Jitter
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**

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.

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. Most improved model highlighted in bold.

		Traditional				Ensembles		
	# classes (% data) removed	VGG-19	Resnet -50	Efficient-Net B3	Efficient-Net B5	PCA	Auto-encoder	Inv. GAN w/ Jitter
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**

5.2.2 Synthetic dataset augmentation

Improvements in performance to single-class ensembles 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

The objective of this study was to compare existing single-class classifier ensembles to traditional classifiers, demonstrate other benefits to using such ensembles besides classification accuracy, and

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				Autoencoder			
	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**

introduce new single-class classification techniques. To accomplish this, several traditional classifiers including VGG-19, Resnet-50, and EfficientNet were trained and compared to ensembles of PCA- and AutoEncoder-based single-class classifiers, as were described in previous literature. As well, a GAN-based approach to single-class classification was implemented by using a StyleGAN2 architecture. After training the generator of a StyleGAN to generate high quality synthetic images of a class, an inversion network is trained to invert images to their latent representation used to generate them. In this way, an effectively larger autoencoder can be trained than previous, allowing for a more sophisticated network. Using reconstruction error as a metric for class likeness in this way improves upon the accuracy in previous models. As well, if Gaussian noise is added to the latent representations of these images during the stage of generation where fine features are generated and this noise is resampled multiple times to form a vote, then classification accuracy is improved even more. We show that the discriminator of these GANs can also be used as single-class classifiers. In one method, we save discriminators after set intervals during the training process and evaluate an image on all of them. The standard deviation of the scores obtained can be used as a metric for class likeness, although a classifier constructed using ensembles of this method do not outperform prior methods. In another method, images are passed through the inversion network and then evaluated by the discriminator. The score output by the discriminator can be used as a metric for class likeness, and classifiers constructed from ensembles of this method do outperform prior methods, but not as much as just the inversion networks with reconstruction loss. We also demonstrated the usefulness of ensembles of single-class classifiers compared to traditional classifiers in flexibility by classifying minimally altered class sets, such as by adding or removing classes. StyleGAN approaches were also shown to be able to increase the classification performance of other models through the use of synthetic data augmentation. Finally, we were also able to improve the rigor of benchmarks on the

PlantVillage dataset by introducing benchmarks with 5-fold cross-validation.

Although many of the methods discussed in this study do not outperform traditional classification techniques in accuracy, the gap between techniques is narrowing. With more sophisticated techniques, it is possible to be able to match or exceed traditional classification performance in addition to model flexibility. There are several avenues for future work to proceed in this direction. First, the ability of diffusion models to be used as single-class classifiers in as similar way as GANs can be investigated [13]. Recently, there have been many advancement in diffusion models, and many image generation and anomaly detection techniques have started switching to them, as they are more robust [36]. A diffusion model could easily see major improvements to the accuracy of the GAN-based models discussed in this study, especially since they already include something similar to the noise-adding technique used to improve GAN classification performance as part of their main construction. Secondly, there can be some research into alternative loss and reconstruction error metrics used in several of the models discussed in this study. MAE and RMSE losses prioritise coarse features over fine features, as they contribute more pixels to the overall image, despite that much classification power in many datasets comes from fine features. A localized reconstruction error could be used instead to prioritise these fine features and improve classification performance. Thirdly, a heterogenous ensemble of techniques described in this study, such as described by Kang, Cho, and Kang [17], could potentially provide even better results. Fourthly, a method of normalizing the likeness scores between classes should be investigated, so that the classifier isn't biased towards classes with abnormally high likeness scores.

Bibliography

[1] Ümit Atila et al. "Plant leaf disease classification using EfficientNet deep learning model." In: *Ecological Informatics* 61 (2021), p. 101182.

[2] Tao Ban and S. Abe. "Implementing Multi-class Classifiers by One-class Classification Methods." In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. 2006, pp. 327–332. DOI: 10.1109/IJCNN.2006.246699.

[3] Christopher Bowles et al. *GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks*. 2018. arXiv: 1810.10863 [cs.CV].

[4] Jinghui Chen et al. "Outlier detection with autoencoder ensembles." In: *Proceedings of the 2017 SIAM international conference on data mining*. SIAM. 2017, pp. 90–98.

[5] Antonia Creswell and Anil Anthony Bharath. "Inverting the Generator of a Generative Adversarial Network." In: *IEEE Transactions on Neural Networks and Learning Systems* 30.7 (2019), pp. 1967–1974. DOI: 10.1109/TNNLS.2018.2875194.

[6] Sarvesh Dubey. *Alzheimer's Dataset (4 class of Images)*. https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images. Kaggle, 2019.

[7] Dave Epstein et al. "Blobgan: Spatially disentangled scene representations." In: *European Conference on Computer Vision*. Springer. 2022, pp. 616–635.

[8] Maayan Frid-Adar et al. "Synthetic data augmentation using GAN for improved liver lesion classification." In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018, pp. 289–293. DOI: 10.1109/ISBI.2018.8363576.

[9] Kemilly Dearo Garcia et al. "An ensemble of autonomous auto-encoders for human activity recognition." In: *Neurocomputing* 439 (2021), pp. 271–280. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.01.125. URL: https://www.sciencedirect.com/science/article/pii/S0925231221001454.

[10] Pei-Yi Hao, Jung-Hsien Chiang, and Yen-Hsiu Lin. "A new maximal-margin spherical-structured multi-class support vector machine." In: *Applied Intelligence* 30 (2009), pp. 98–111.

[11] Kaiming He et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[12] M.A. Hearst et al. "Support vector machines." In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28. DOI: 10.1109/5254.708428.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[14] David Hughes, Marcel Salathé, et al. "An open access repository of images on plant health to enable the development of mobile disease diagnostics." In: *arXiv preprint arXiv:1511.08060* (2015).

[15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[16] Piotr Juszczak and Robert P. W. Duin. "Combining One-Class Classifiers to Classify Missing Data." In: *Multiple Classifier Systems*. Ed. by Fabio Roli, Josef Kittler, and Terry Windeatt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 92–101. ISBN: 978-3-540-25966-4.

[17] Seokho Kang, Sungzoon Cho, and Pilsung Kang. "Multi-class classification via heterogeneous ensemble of one-class classifiers." In: *Engineering Applications of Artificial Intelligence* 43 (2015), pp. 35–43. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2015.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0952197615000846.

[18] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[19] Tero Karras et al. "Analyzing and improving the image quality of stylegan." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.

[20] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation." In: *arXiv preprint arXiv:1710.10196* (2017).

[21] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[22] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114* (2013).

[23] Daewon Lee and Jaewook Lee. "Domain described support vector classifier for multi-classification problems." In: *Pattern Recognition* 40.1 (2007), pp. 41–51.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[25] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. "Rectifier nonlinearities improve neural network acoustic models." In: *Proc. icml*. Vol. 30. 1. Atlanta, GA. 2013, p. 3.

[26] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[27] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "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*. Springer. 2015, pp. 234–241.

[29] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[30] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[31] David M.J Tax and Robert P.W Duin. "Support vector domain description." In: *Pattern Recognition Letters* 20.11 (1999), pp. 1191–1199. ISSN: 0167-8655. DOI: `https://doi.org/10.1016/S0167-8655(99)00087-2`. URL: `https://www.sciencedirect.com/science/article/pii/S0167865599000872`.

[32] David MJ Tax and Robert PW Duin. "Growing a multi-class classifier with a reject option." In: *Pattern Recognition Letters* 29.10 (2008), pp. 1565–1570.

[33] M.A. Turk and A.P. Pentland. "Face recognition using eigenfaces." In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1991, pp. 586–591. DOI: `10.1109/CVPR.1991.139758`.

[34] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. "A survey of transfer learning." In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[35] Cort J Willmott and Kenji Matsuura. "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance." In: *Climate research* 30.1 (2005), pp. 79–82.

[36] Julia Wolleb et al. "Diffusion models for medical anomaly detection." In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2022, pp. 35–45.

[37] Eric Wu et al. "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*. Springer. 2018, pp. 98–106.

[38] Weihao Xia et al. *GAN Inversion: A Survey*. 2022. arXiv: `2101.05278 [cs.CV]`.

[39] Kaichao You et al. "How does learning rate decay help modern neural networks?" In: *arXiv preprint arXiv:1908.01878* (2019).

[40] Jun-Yan Zhu et al. "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*. Springer. 2016, pp. 597–613.

[41] Yin Zhu et al. "Heterogeneous Transfer Learning for Image Classification." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 25.1 (Aug. 2011), pp. 1304–1309. DOI: `10.1609/aaai.v25i1.8090`. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/8090`.