Open-World Classification using Adversarial AutoEncoders

Sudhanshu Mishra

Indian Institute of Technology, Kanpur 208016

Prof. Nikhil R. Pal

Indian Statistical Institute, Kolkata 700108

Abstract

Recent advances in research have established Multi-Layered-Perceptron (MLPs) as robust classifiers. However, MLPs suffer from a very fundamental problem where they may classify a test data point which does not belong to any of the classes to one of the given classes. The MLP's prediction for points that are outside the training window is very erratic and strange. This happens mainly because most classifiers are trained using the closed-set assumption. Real World Classification problems mostly fall under the open-set assumption, where it is not uncommon for the classifier to encounter data which does not belong to any of the given classes. Hence, the classifier should be able to classify these data points into an "unknown" class. Generative adversarial network (GAN) is the most exciting machine learning breakthrough in recent years, and it trains the learning model by finding the Nash equilibrium of a two-player zero-sum game. During training, the GAN tries to learn the sampling window using the training data and then predicts two scores (G-Score and D-Score) to estimate whether the given data belongs to an unknown class or to one of the given classes. We propose an Adversarial Autoencoder approach to distinguish open world samples from the training samples using the reconstruction error as a viable metric.

**Keywords**: GAN, Auto-encoders, Open-Space Risk, Adversarial Loss, Classification

**Link to the code** : <https://github.com/ghostktjMactavish/Adversarial-Autoencoders>

Abbreviations

|  |  |
| --- | --- |
| SRF | Summer Research Fellowship |
| IAS | Indian Academy of Science |
| MLP | Multi Layer Perceptron |
| AAE | Adversarial Auto-Encoder |
| GAN | Generative Adversarial Network |
| V(G,D) | Value Function of a GAN |
| G | Generator |
| D | Discriminator |
| EVT | Extreme Value Theory |

1 INTRODUCTION

In recent years there have been many breakthroughs in the field of deep learning. Multi-Layer Perceptron (MLP’s) with a single hidden layer have been used to solve numerous problems in a variety of domains. It has been proved that MLP’s act as universal approximators for a large class of nonlinear functions. An important application of MLP’s is in the problems related to classification. MLP’s in general act as a good classifiers, even surpassing human abilities in a number of domains. Despite their numerous advantages MLP’s suffer from non-interpretability. This lack of readability often limits the use of MLP’s on critical applications such as diagnosis of cancer.

Most classifiers work very well under the closed-set assumption. The closed set assumption states that any incoming data to the classifier always belongs to one of the given classes. However, most real world problems do not satisfy the closed set assumption. It is not uncommon for a classifier to receive data that does not belong to any of the known classes. These types of problems are called open-world classification problems. People generally rely on the output of the MLP for any data point without taking into consideration the position of the data with respect to the training data. The response of an MLP for points well outside the “boundary” is very erratic and hence unreliable. For example, for points well outside the convex hull of the training data, the output of the MLP could be very high and completely useless. In critical applications like medical diagnosis and non-destructive testing (NDT) this may lead to very serious problems. Therefore, it is very important for an MLP to identify such data points and handle them accordingly. The MLP should be able to classify such points into a new class called the unknown class.

The idea of auto-encoders has been part of the historical landscape of neural networks for decades. Traditionally, auto-encoders were used for dimensionality reduction or feature learning. Recently, theoretical connections between auto-encoders and latent variable models have brought auto-encoders to the forefront of generative modeling. Under-complete auto-encoders with code dimensions less than input dimensions, can learn the most salient features of the data distribution. Choosing the code dimensions and the capacity of the encoder and decoder based on the complexity of the distribution to be modeled. Regularized Auto-encoders use a loss that encourages a model to have other properties besides just copying the input to its output. In the given approach we try to use these properties of regularized auto-encoders using an adversarial auto-encoder to detect data that does not belong to the training distribution and comes from the open space.

The auto-encoder learns some intrinsic properties about the data it is trained on as it tries to map it into the given latent space. During the testing phase when unknown data from the open space is fed to the auto-encoder the reconstructed output looks similar to the given training data. While analyzing the reconstruction error of the auto-encoder, we saw that there was an appreciable difference between the errors for the known and unknown data.

Extra work may be done by trying other variants of the adversarial auto-encoder by incorporating label information, as mentioned in [9]. Using additional information in the adversarial training might enhance the distinction between the known classes and unknown classes. Some work has to be done to see how the adversarial auto-encoder defines the open space similar to how it is defined in [1].​

1.1 Statement of the Problems

Any MLP used as a classifier suffers from the open world problem, i.e. it tries to place data that does not belong to any of the known classes into one of them. As the MLP has not seen any data from an unknown class during training hence it does not know how to handle such data. Thus, each MLP must be able to distinguish open world data from known data.

We propose an adversarial auto-encoder architecture which can generate a reconstruction score that can be used to distinguish known data from unknown data. We further provide a paradigm to tune the threshold, to mark known and unknown samples for a particular task.

2. LITERATURE REVIEW

2.1 The Open World Problem

Classic supervised learning techniques make the closed world assumption meaning that all the classes that it will see during testing will be from one of the classes on which it is trained on. Although this assumption holds in many applications, it is violated from many others especially in dynamic and open environments. In these environments data may contain instances from classes that have not yet appeared in training. To learn in such environments we need open-world classification, which can detect instances of unseen classes in testing or model application. Open world learning can be broadly defined as learning a model that can perform its intended task and can also identify new things that have not been learned before, and then incrementally learn new things. For example in reading, the system may see a new word that it does not know and then it learns it by looking into a dictionary. Open-world learning basically performs a form of self-motivated learning because by recognizing that something new has appeared, the system has the opportunity to learn new things. This type of system has the curiosity which motivates it to learn new things. In the context of supervised learning, the key is for the system to recognize what it has not seen or learned before. If a learned model cannot recognize anything new, there is no way for the learner to learn new things or to explore by itself other than by being told or instructed by a human user or an external system, which is not ideal for a truly intelligent system. It also has great difficulty to function in a dynamic and open environment. In the field of computer vision it is known as open-world recognition [2].

2.2 Towards Open Set Recognition

In the majority of deep networks the output of the last fully-connected layer is fed to the SoftMax function, which produces a probability distribution over the N known class labels. While a deep network will always have a most likely class, on may hope that for an unknown input all the classes would get a low probability and thresholding on uncertainty would reject unknown classes. Recent papers have shown how to produce “fooling” or rubbish images that are visually far from the desired class but get a high probability score. They strongly suggest that thresholding on uncertainty is not sufficient to determine what is unknown. By extending deep networks to threshold Softmax probability improves open set recognition but does not resolve the issue of fooling images. [3]

While fooling/rubbish images are, to human observers, clearly not from a class of interest, adversarial images present a more difficult challenge. These adversarial images are visually indistinguishable from a training sample but are designed so that deep networks produce high confidence but incorrect answers. This is different from standard open space risk because adversarial images are near a training sample in input space, for any given output class. A key insight that they noticed in their deep networks was that open space risk should be measured in feature space rather than pixel space. [2]

2.3 Generating Data from the Open Space

Data points from a pattern class usually will be generated following some distribution. The distribution function can give information about the spatial boundary of the pattern class. But in most cases it is very difficult to estimate a prior distribution from the data points of a class. One way to determine the class boundary is to estimate the sampling window from which the points from that class are generated. The convex hull of the points in the training data may be a good measure of the boundary of the training data. The smallest rectangle containing the data was taken as the boundary of the class. Then the edges were increased by 5 % on each side. They then generated 256 x 256 points uniformly covering the entire rectangle. They considered a point to be classified as a class if the output of the kth output node is more than 0.8 and the output of all other nodes is less than 0.2. Although the training set showed zero misclassification the generalization was very poor.

|  |
| --- |
|  |

Fig.1The open world problem in action. The colored symbols are the classes on which the classifier was   
trained.We can see even far away from the training data in the bottom right area, the classifier assigns the data to the red circle class.

The figures given in the paper are similar to Fig. 1, showing that the MLP performs quite well on the training data and also on the points that lie inside the boundary of the training data but its response on points outside the training data does not follow any specific behavior and appears to be very strange. By using the given strategy they were able to get strict generalization inside the hypercube containing the dataset. [4]

They used a very naive approach for the definition of the boundary of a class. It works well for points inside the hypercube surrounding the dataset but far from it may still suffer from the same problems as any other classifier. This is because the model itself is a regular MLP or rather a group of MLPs merged into one, just the training of the model helps it in identifying the unknown points inside the hypercube. It still suffers from the same problem as any other classifier for points outside the hypercube and so it cannot be used for such points.​[5]

2.4 Auto-encoders

An auto-encoder is a neural network that attempts to copy its input to its output. It mainly consists of two parts, the encoder and the decoder. Internally it consists of a hidden layer that describes a code used to represent the input. Auto-encoders are trained to copy only approximately, and copy only input that resembles the training data. As the model is forced to prioritize which aspects of the input should be copied it learns useful properties of the data. One way to obtain useful features from the auto-encoder is to constrain the latent space or code to have a smaller size than the data. Such an auto-encoder is called an under-complete auto-encoder. Learning an under-complete representation forces the auto-encoder to learn salient features of the data. The learning process is described simply as minimizing a loss function L(x,g(f(x))) where g represents the decoder and f represents the encoder. L is a loss function that penalizes g(f(x)) for being dissimilar to x. It is usually taken to be the mean squared error (MSE). Regularized auto-encoders use a loss function that encourages a model to have other properties besides just copying the input to the output. These other properties include sparsity of the representation, smallness of the derivative of the representation, and robustness to noise or to missing inputs.​[6]

2.5 Generative Adversarial Networks

The promise of deep learning was to find rich theoretical models that could represent the probability distributions normally encountered in artificial intelligence applications like Natural Language Processing, images, audio waveforms representing speech signals, etc. The most striking successes in deep learning so far has been in discriminative models that map a rich high dimensional input to a class label. Generative Adversarial Networks [7], are a new type of generative models that overcome all the difficulties faced by previous generations of generative models.

In GANs, the generator or the generative model is pitted against an adversary, called the discriminator, that learns to predict whether a given sample comes from the data distribution or the model distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect counterfeit currency. This form of adversarial training tends to improve both models, resulting in a generator that produces data from a distribution that is indistinguishable from the actual data distribution. The end result is that we get a generator that maps a latent space to a high dimensional space that is similar to the data distribution. On the other hand, we also get a discriminator that is able to distinguish between real samples, samples that come from the data distribution, and fake samples, samples that are not from the data distribution.​​

2.6 Adversarial Auto-encoders

The adversarial auto-encoder [9] is a new approach that can turn an auto-encoder into a generative model. The adversarial auto-encoder is trained with dual objectives. One is minimization of the traditional reconstruction error of an auto-encoder while the other is the adversarial loss, as done in GAN [7]. The adversarial training helps match the posterior distribution of the latent space to an arbitrary prior distribution. This adversarial training has a strong connection to the Variational Auto-encoder training. The result of the training is that the encoder learns to convert the data distribution to the prior distribution, while the decoder learns a deep generative model that maps imposed prior to the data distribution.

The adversarial training acts as a regularizer to the auto-encoder by matching the aggregated posterior of the latent space to the assumed prior distribution. In order to do so an adversarial network is attached in parallel to the encoder of the auto-encoder. The generator of this adversarial network is the encoder of the auto-encoder and an additional discriminator which completes the adversarial network. The encoder tries to fool the discriminator that the aggregated posterior comes from the assumed prior distribution. This in turn helps in shaping the latent space into any assumed prior distribution. The auto-encoder tries to minimize the reconstruction error between the input and the output as is done in a standard auto-encoder [6​].

2.7 Open World Classification using GANs

Initially GANs were used for image generation to help augment the dataset. Since its inception in 2014 it has been successfully applied to a wide range of applications like image super-resolution, image segmentation, image detection, image in-painting, and image de-occlusion. Applications of the GAN model have been extended to video generation, encryption and decryption, 3D model, text generation, machine translation, and drug development. Different variants of GAN, such as least squares GAN, energy based GAN, Wasserstein GAN and boundary equilibrium GAN. As a new kind of generative model, GAN also gains attention in dealing with classical machine learning problems such as clustering, unsupervised feature learning, classification, transfer learning, ensemble learning and reinforcement learning.

The ability of the generator in GANs to estimate the data distribution and that of the discriminator to distinguish between real and fake data can be used to effectively solve the Open World Classification problem. In this approach the GAN is fed the known data as the training data. This in turn results in the generator learning a mapping from the latent space to the known data distribution while the discriminator learns to identify the known data distribution. During testing when data from the open world is encountered, the latent space of the generator is searched to find the latent space representation that upon generation produces the minimum reconstruction error between the output of the generator (for that latent variables) and the open world data [10]. This reconstruction error can serve as a good score to identify whether the open world samples are from the known classes of not. Furthermore, the discriminator score for the test data can also be used as a good metric for the identification of open world samples. The discriminator upon learning the distribution for the known data tends to call the open world samples as fake. This makes the output of the discriminator a good metric for identifying open world samples.

The generator and discriminator scores for a data point can serve as good indicators for the identification of open world samples. The search for the minimum reconstruction error in the latent space is quite time consuming. We propose to reduce this time constraint by using an auto-encoder for the minimization task.

3. METHODOLOGY

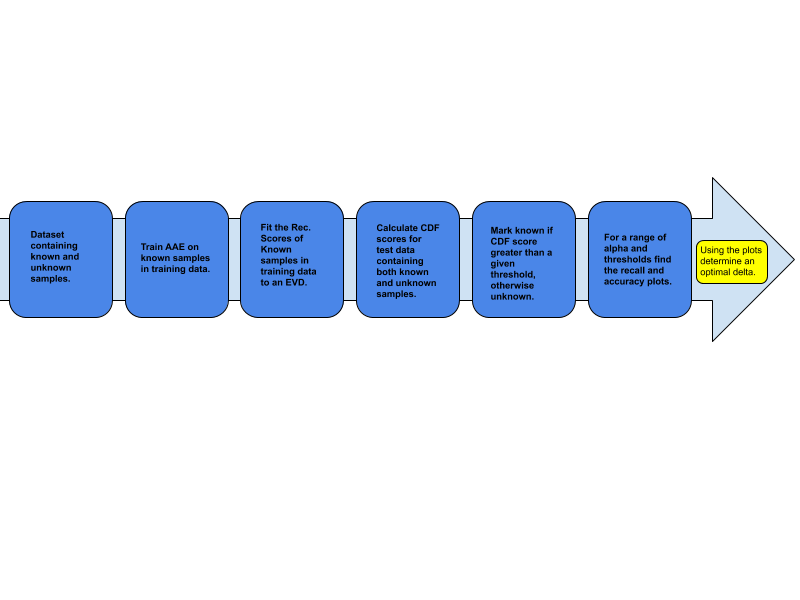


Fig 2. Flowchart showing the methodology used.

Some terms that will be used ahead:

1. **alpha** : fraction of unknown samples in custom dataset
2. **threshold** : the CDF value above which a sample is labelled unknown.
3. **delta** : the final threshold value decided based on the experiments

We start with a dataset having a set of known samples and unknown samples. We train our AAE on the known samples present in the training data. Using the trained AAE we calculate the reconstruction scores for the samples in the training and test dataset, both known and unknown samples. Using the reconstruction errors of the known samples in the training data, we fit these errors to an extreme value distribution (EVD). Using the obtained EVD, we calculate the CDF scores of the reconstruction errors of the test samples. We then create a sample dataset by randomly choosing 5000 samples from the test data. We create multiple such datasets, each having a different alpha. For a given threshold, we label the CDF score as known if it is less than the threshold value, otherwise we label it as unknown. For each alpha, we plot the recall and accuracy for a range of threshold values. Using these plots, one can decide upon an optimal delta to use for their specific problem.

3.1 Using the Generative Adversarial Networks to identify unknown classes

The approach we took to solve this problem was similar to that used in [10]. The given training set contains samples from only the known classes while the test set contains samples from both the known and unknown classes. In their approach they used a GAN to estimate the training data distribution.

Their architecture consisted of a Vanilla GAN having a generator G(z) and a discriminator D(x). Both the generator and discriminator are MLPs. The input to the generator is random noise. The generator takes this noise as input and outputs data whose dimensions are equal to the dimensions of the training data. The input to the discriminator has the same dimensions as the training data. The discriminator is fed samples from both the training set as well as the data generated by the generator. It outputs a single value that lies between 0 and 1. This value signifies the discriminators confidence in saying whether the data is from the training distribution or not. The generator consists of two dense layers each containing 1000 nodes which are finally connected to another dense layer having 784 nodes. The initial two layers both have LeakyRelu activation functions while the output layer has a tanh activation function. The discriminator also has 2 dense layers connected to its input followed by a single node as its output. The dense layers have LeakyRelu activation function while the output of the discriminator has a sigmoid activation.

Both the generator and the discriminator are trained simultaneously using SGD. At the end of the training the generator is expected to have learnt a mapping from the random noise distribution to the training distribution while the discriminator is expected to have learnt to distinguish between data that belongs to the training distribution and data that does not belong to the training distribution. During training, both D and G play a two-player minimax game with a value function V(G,D). The generator tries to minimize V while the discriminator tries to maximize V.

|  |
| --- |
|  |

Fig 3 The value function in GANs[9].

Inorder to identify whether a data instance is from the training distribution or not they repeatedly sample through the latent space to minimize the following loss function:

|  |
| --- |
|  |

Fig 4 The reconstruction loss used in the study.

Where x refers to the testing data and z refers to points in the random noise, ‘s’ signifies the dimension of the testing data. The minimized reconstruction error is taken as the G-Score and while the output of the discriminator is taken as the D-Score. For points belonging to the training distribution the G-Score is expected to have a small value while the D-Score has a high value. For data belonging to the Open Space the G-Score is expected to have a large value while the D-Score is expected to have a low value. These G-Score and D-Score values help us in identifying whether the point comes from the training distribution or from the Open Space.

3.2 Using Adversarial Auto-encoders to identify unknown classes

Using the same concept of minimizing the reconstruction error we propose to identify unknown classes using the representation learning properties of the adversarial auto-encoder. The auto-encoder is an MLP that learns some intrinsic properties of the data that it is trained on. In a standard auto-encoder the input is passed to an encoder, which is also an MLP that maps it to a lower dimensional space. This lower dimensional data is then passed to the decoder which tries to reconstruct the original input by minimizing the reconstruction error, which is in most cases just the mean squared error. However, the standard auto-encoder does not generalize well. Furthermore, similar type of data may not be mapped close to each other in the latent space. To overcome these disadvantages an adversarial auto-encoder also adds a regularizer in the form of an adversarial loss [7]. The adversarial training helps in shaping the latent space into a known prior distribution. This helps in generalizing the auto-encoder learning. Furthermore similar data are mapped close to each other. We propose to use the adversarial auto-encoder to identify unknown classes or data from the open world. In this scheme we can avoid the repeated sampling of latent space to minimize the reconstruction loss function[.





Fig 5. The loss functions used in the Adversarial Auto-encoder.

In the above equation[9], G is the encoder and D is the Discriminator of the Adversarial Auto-encoder.

Similar to the approach mentioned in the previous section, the dataset is divided into training and test datasets. The training set contains data from the known classes whereas the test set contains data from both the known and unknown classes. The AAE architecture is similar to the architecture used in [9]. During training the training set is fed to the AAE. The AAE consists of 3 components, namely the encoder, the decoder and the discriminator. The encoder is also the generator of the adversarial training involved. The encoder consists of 2 dense layers containing 1000 nodes each. Each dense layer has a Leaky Relu activation function. Finally it is connected to a linear layer whose dimensions are smaller than the data dimension. In our experiments we have kept the dimension of the latent space as 8. However, this is a hyperparameter of the AAE architecture and can be tuned further to obtain better results. The encoder receives the input data and maps it into the latent space. This latent representation is then simultaneously passed on to the decoder and the discriminator. The discriminator consists of 2 dense layers with a Leaky Relu activation function followed by a single node that has a sigmoidal activation function. The discriminator receives the output of the encoder as the input for which it is trained to output a 0 whereas when it receives input from the prior distribution it is trained to output a 1. The encoder and the discriminator are trained in an adversarial manner similar to how a GAN is trained. The decoder also consists of 2 dense layers with a Leaky Relu activation function followed by another dense layer whose dimensions are equal to that of the input data of the AAE. This output layer has a sigmoidal activation function.

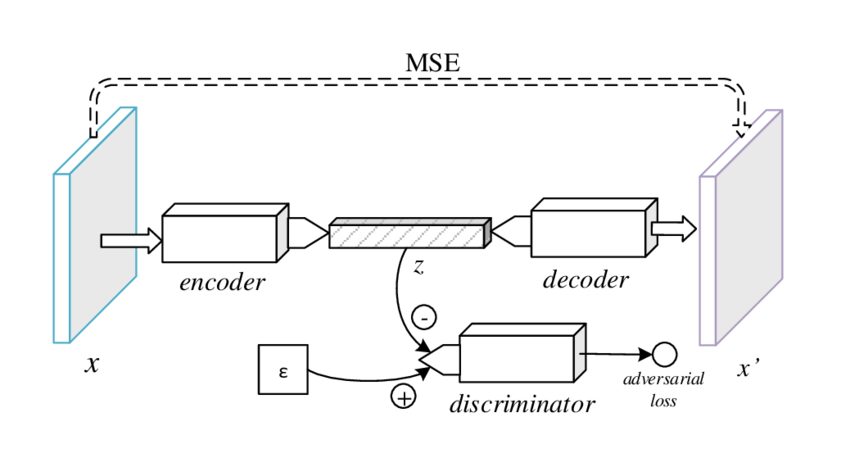
****

Fig 6. The adversarial auto-encoder architecture used in the study.

The training of the AAE involves two loss functions, namely the reconstruction loss and the adversarial loss. The reconstruction loss is just the mean squared loss between the input and the output of the decoder of the AAE. Minimizing this loss helps the AAE to get better reconstruction of the input data. The adversarial loss is similar to the GAN loss function. Minimizing this loss helps in imposing the prior distribution on the latent space. This loss acts as a regularizer that constrains the latent space to a given prior distribution.

Using this trained AAE we are going to distinguish between the known and the unknown classes. During the testing phase we supply the test set to the AAE, which contains both the known and the unknown classes. We then observe the reconstruction error for each data. As the AAE has been trained for the known classes, the reconstruction error for such data is small. While when we feed unknown classes to the AAE, the encoder still tries to map this data to the same latent space and due to the regularizing effect of the adversarial training it is mapped close to the points belonging to the known data. Thus when the decoder tries to reconstruct the unknown classes it is unable to do so resulting in a much higher reconstruction error for the unknown data. Thus, we may be able to identify the known and unknown data based on their reconstruction errors. The reconstruction error for the known classes is expected to be quite small while for the unknown classes it is expected to be much higher.

This method of using the minimized reconstruction error is similar to that used in the approach mentioned above but instead of minimizing the error using an iterative method like Gradient Descent, it uses the representation learning abilities of the AAE. Thus, computationally it is much more efficient.

3.3 Using the Extreme Value Distributions to predict open world samples

Most of the time we deal with problems where the mean or the average is a good representation of the data. But in this problem, encountering unknown samples is an extreme case and so the mean or average does not give us relevant information about the data. Therefore, in this case we use the extreme value theory and the extreme value distributions to get insight into the data.

The extreme value theory deals with the distribution of the extreme values (like the maximum or the minimum) from the (i.i.d.) samples from some distribution. These values converge to a class of distributions called the generalized extreme value distributions which characterize three types of distribution namely Weibull, Gumbel and Frechet distributions. These distributions help us quantify the probability of observing extreme values. In the case of maximum values, this probability is computed by finding the cumulative density functions. By thresholding the cumulative density function, we can identify values that occur very rarely.

So, using the reconstruction errors of the training data, we fit them to an extreme value distribution. The reconstruction scores are fit to an extreme value distribution using the extreme value distributions present in the *stats* module from the *scipy* package[12]. From the stats module we use the genextreme function to fit the data to an EVD. Using this distribution, we calculate a CDF score for the test data, both for the known and unknown classes. We now decide on a threshold using which we can classify the known and unknown classes. For a CDF score which is greater than the threshold we classify it as an unknown sample, otherwise we classify it as a known sample. As the probability to get an unknown sample is quite high, the corresponding CDF score for the unknown sample will also be quite high.

Deciding this threshold without the actual scenario is quite hard and depends a lot on the problem at hand. We defined a parameter alpha, which denotes the fraction of unknown samples in the data. For a series of values of alpha, we find the recall and accuracy of each class with respect to a range of thresholds. We present plots showing the trend in recall for each class with respect to a series of thresholds for each alpha. Using these plots, a threshold value can be decided which works best for the problem at hand. We also present the recall and corresponding threshold value for which the recall for both the classes is the highest. This value of threshold may not be optimal for all cases, as in problems where detecting the unknown class is more important a different value of the threshold can be chosen for which the recall of the unknown classes is higher sacrificing on the recall of the known classes.

4 RESULTS AND DISCUSSION

All experiments were performed on the MNIST dataset[13]. We divided the experiments into 3 types. In the first type of experiment we took one of the digits as the known class while the other as the unknown class. We trained an AAE on this known class. During testing we fed the AAE data from both known and unknown class.

We took each of the digits as the known class and observed the reconstruction error in each case. In the second type of experiments we took all possible pairs of digits as the known class and the rest as the unknown class. We looked at the reconstruction error for each case in these experiments. In the final type of experiments we randomly took 4 classes as the known class and the rest as the unknown class. We did not take all possible combinations due to time constraints.

4.1 One Known Class

In this experiment we took one of the digits as the known class and fed it to the Adversarial Auto-encoder during the training stage. We performed a total of 10 experiments in which we took one of the digits as the known class. We present the results obtained by taking the digit '0' as the known class. Here are the loss curves observed during training for the digit '0':

|  |  |
| --- | --- |
|  |  |
| Fig 7The adversarial loss curve for Known Class '0 | Fig 8 The reconstruction loss curve for Known Class '0' |

|  |
| --- |
|  |

We present a few of the original and corresponding reconstructed images of the training data after the training was completed in Figs 6 and 7 respectively..

|  |  |
| --- | --- |
|  |  |
| Fig 9 The original images | Fig 10 The reconstructed images |

We see that the imposing of the prior distribution on the latent space was successful by the adversarial training. Some samples from the prior distribution result in the images shown in Figure 11.

|  |
| --- |
|  |

Fig 11 Images constructed by the decoder while sampling from the imposed prior distribution

Fig. 9 depicts some images from the unknown or open world classes that were submitted to the system. Fig 10 shows the corresponding decoded images. From Fig. 10, we see that the decoder still tries to reconstruct images that look similar to the original data, Fig. 8.

|  |
| --- |
|  |

Fig 12 Original images of the unknown classes

|  |
| --- |
|  |

Fig 13 Reconstructed images of the unknown classes

This shows that the open world samples are also imposed to a region in the latent space which is similar to that of the known samples which results in the decoder constructing images that look like the known samples. This behavior of the AAE results in a large reconstruction error for the open world data while a small reconstruction error for the known data. The following histograms show the reconstruction error obtained on data from each of the classes. The green plot is the test data of the known class, in this case the digit '0', while the red plots are the unknown classes. All available test instances for the unknown class were used for obtaining the histograms.

|  |
| --- |
|  |

Fig 14 Histograms showing the distribution of reconstruction errors for each class with the digit '0' as the known class.

The combined histogram in Figure 12 of the reconstruction errors for the known and unknown data provides a much better visualization of the results.

|  |
| --- |
|  |

Fig 15 The combined histograms of the reconstruction errors of the known and unknown data in this experiment.

Is Table 13 computed for one known class? How do we interpret the entries? For example, the (0,8) entry is 0.07136. The actual input is zero, then what does this entry mean?

We now present the mean reconstruction error obtained for each class in all of the experiments.

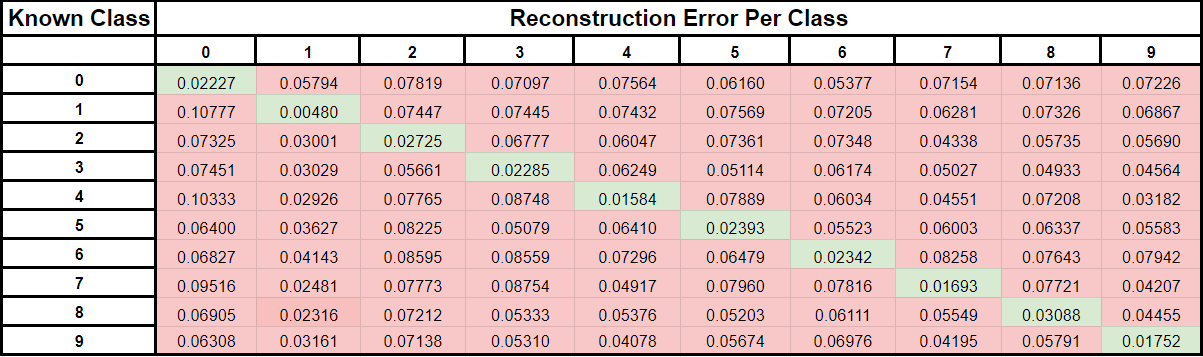


Fig 16 Table containing mean reconstruction error per class for all experiments. Each row is a separate experiment with the columns having the mean reconstruction error for that class in that experiment.

Table 13 contains the mean reconstruction errors obtained in the 1 known class experiments. This table contains the mean reconstruction error obtained for each class when one of the digits is known. Each row represents one experiment where the green cells denote the known class and the red cells denote the unknown class. As can be seen in each experiment the mean reconstruction error for the known class is much lower than the mean reconstruction error of the unknown classes. The errors listed above have been calculated on the test data of the all the digits.

We finally fit the reconstruction errors of the known class(es) to an extreme value distribution [11], thereby obtaining a CDF score for the reconstruction errors of the unknown class(es). In Fig. 14, we fit the EVT to the 10 cases, where in each case only one class is known and the remaining 9 are unknown. We see that the CDF score assigned to most of the unknown data are in the more than 90 percentile region.

|  |
| --- |
|  |

Fig 17 EVT plots showing CDF score vs reconstruction error for the known and unknown data

|  |
| --- |
|  |

Finally, we created sample datasets from the test data for different alphas. This sample dataset has 5000 samples. We then present the plots corresponding to the recall and accuracy for different values of alpha. We also label the recall and threshold and the corresponding accuracy for which the recall on both the known and unknown classes is the highest. Using this plot an appropriate value of threshold can be chosen for the model depending upon the use case.

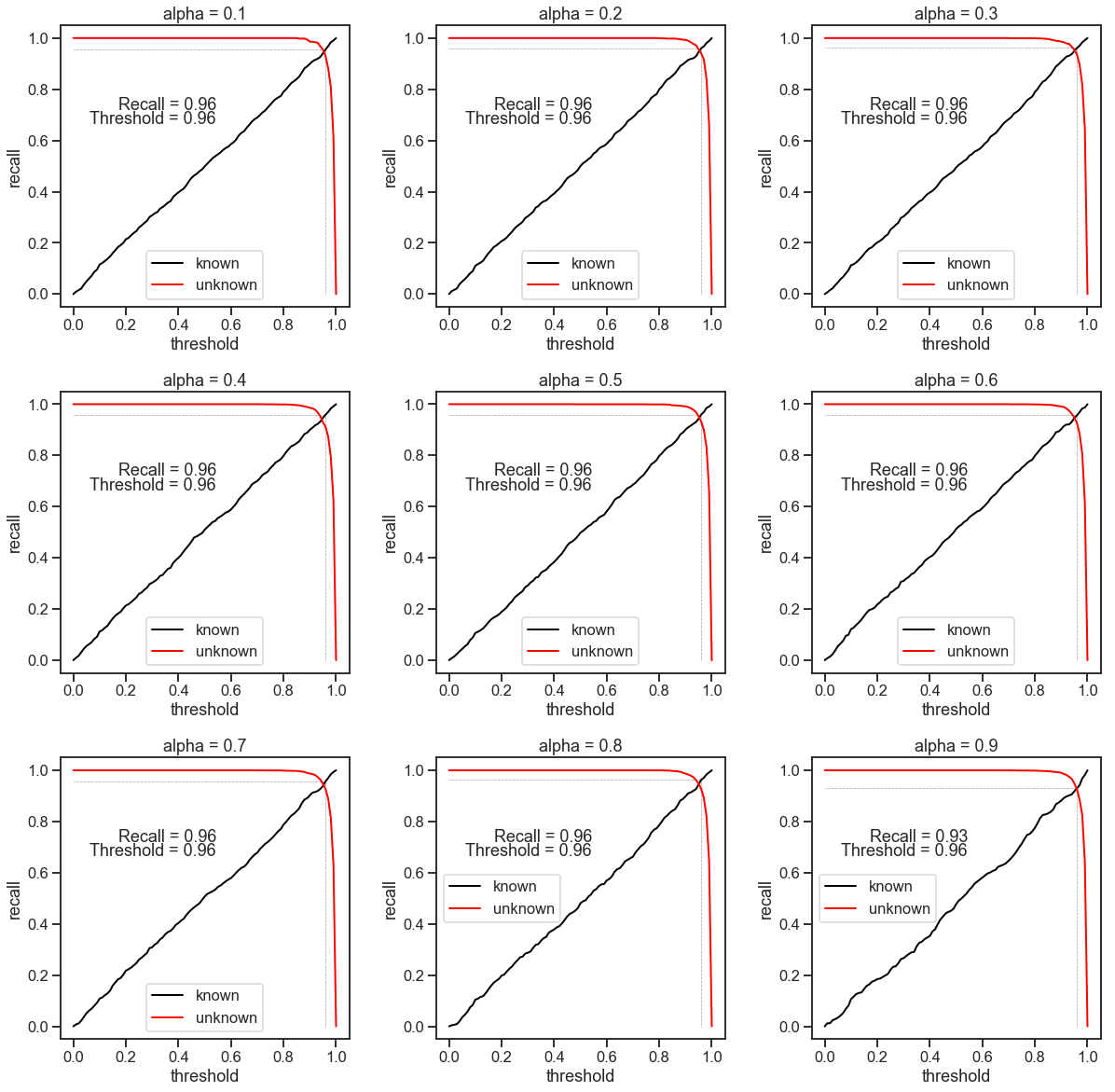


Fig18. Recall and Threshold Plot for different values of alpha for the digit ‘0’.

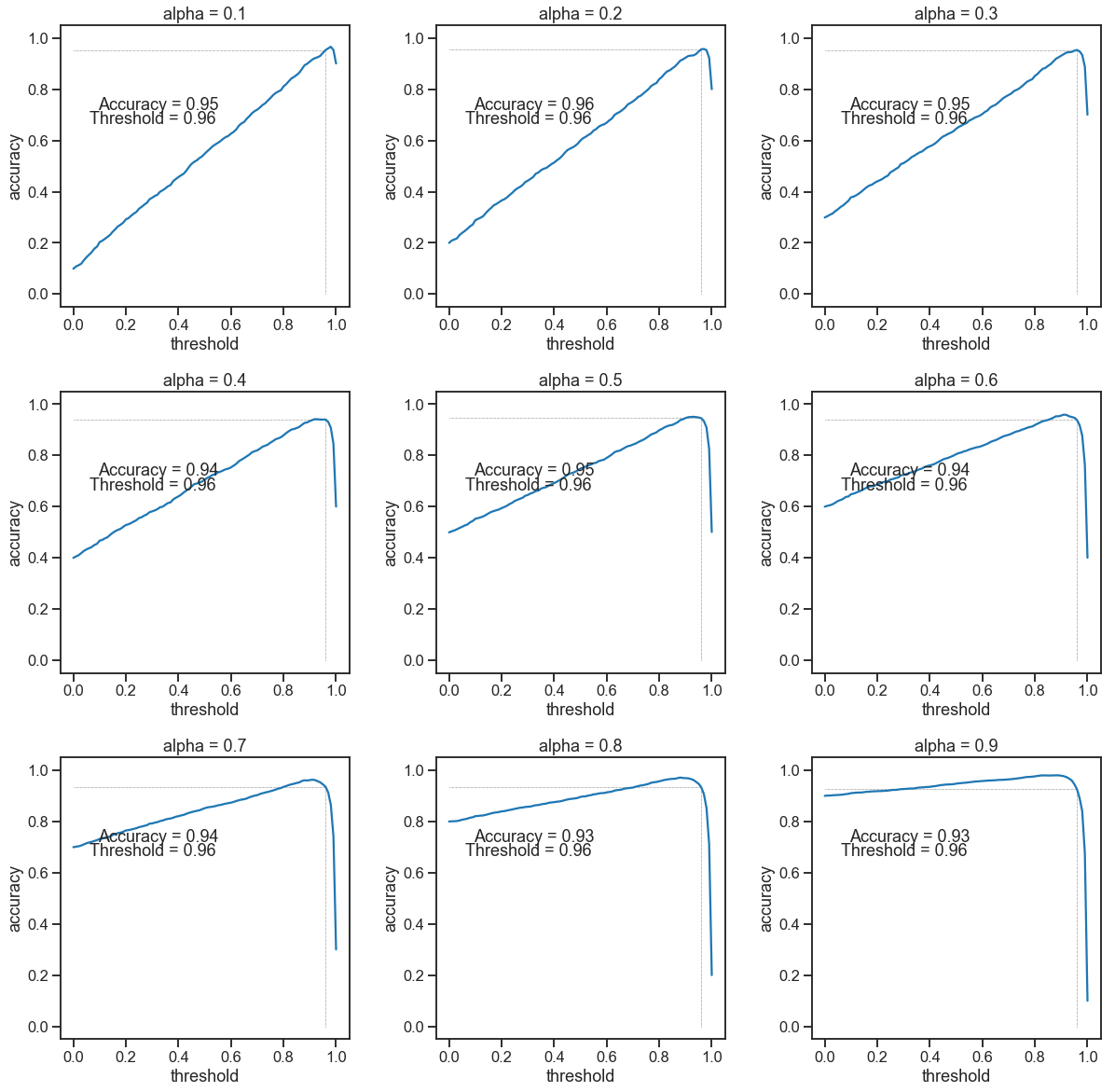


Fig19. Accuracy and Threshold Plot for different values of alpha for the digit ‘0’**.**

Figures 18 and 19 reveal that the accuracy depends on the no. of samples but the recall remains almost constant for different alpha. Thus, recall is a better metric for evaluation in this case. Furthermore, an alpha corresponding to 0.1 and 0.2 is much more realistic as unknown samples are quite rare. Thus, the plots for these alphas should be given more importance while deciding the value of the threshold.

Comments: 1. Please introduce CVT before you use it. Explain here how you estimate the parameters of the EVT. Explain the decision rule that you like to use and the report the accuracy on the trained classes as well as % of test data (from the complement classes) that are rejected by your system.

4.2 Two Known Classes

For this experiment we tested our approach on all pairs of digits in the MNIST dataset. For a given pair of digits as the known class we trained the AAE on the given dataset. We performed 45 experiments, taking all possible pairs of digits as the known class and the complement eight classes as the unknown classes. We present the results for the digits '0' and '7' as the known class. Figs. 15 and 16 show the adversarial and reconstruction losses for the experiment with 0 and 7 as known classes.

This is comment is applicable to other experiments also when two known classes or 4 known classes.

|  |  |
| --- | --- |
|  |  |
| Fig 20 Adversarial Loss Curve for '0' and '7' as   known classes | Fig21. Reconstruction loss curve for '0' and '7' as the known classes |

Figs. 17 and 18 depict some of the original and reconstructed images of 0 and 7.

|  |  |
| --- | --- |
|  |  |
| Fig 22 Original '0' and '7' images | Fig 23 Reconstructed '0' and '7' images |

|  |
| --- |
|  |

Fig. 19 displays some of the reconstructed images when the inputs sampled from the prior distribution. The reconstructed images look very much like the known classes.

|  |
| --- |
|  |

Fig 24 Images constructed by the decoder while sampling from the imposed prior distribution

Even when data from the unknown classes are used as inputs (Fig 20), the reconstructed images look samples from the known classes and consequently will lead to very high reconstruction errors.

|  |
| --- |
|  |

Fig 25 Original Images of the unknown classes

|  |
| --- |
|  |

Fig 26 Reconstructed Images of the unknown classes

This suggests that that the AAE tries to encode the open world samples into the same region of the latent space as that occupied by the known data. This results in the decoder producing an output that looks very much like the known data. We now present the histograms showing the reconstruction errors for each data for all the classes. This histogram contains '0' and '7' as the known classes.

|  |
| --- |
|  |

Fig 27. Histograms showing the distribution of reconstruction errors per data for each class with the digits ‘0’ and ‘7’ as the known classes.

The combined histogram of the reconstruction errors for the known and unknown data provides a better visualization of the results.

|  |
| --- |
|  |

Fig 28. The combined histograms of the reconstruction errors of the known and unknown data taking ‘0’ and ‘7’ as known classes. Blue plot is for the known data and Orange is for unknown data.

The following table contains the mean reconstruction errors obtained in all of the experiments.

|  |
| --- |
|  |

Fig 29 Table containing mean reconstruction error per class for all experiments. Each row is a separate experiment, with the columns having the mean reconstruction error for that class in that experiment.

Table 24 contains the mean reconstruction errors obtained in the 2 known class experiments. This table contains the mean reconstruction error obtained for each class when two of the digits belong to the known class. Each row represents one experiment where the green cells denote the known class and the red cells denote the unknown class. As can be seen in each experiment the mean reconstruction error for the known class is much lower than the mean reconstruction error of the unknown classes. The errors listed above have been calculated on the test data of the all the digits.

The explanation that I have asked for in connection with your previous similar table is also applicable here.

We finally fit the reconstruction errors of the known class to an extreme value distribution thereby obtaining a CDF score for the reconstruction errors of the unknown class. We see that the CDF score assigned to most of the unknown data is more than 90 percentile.

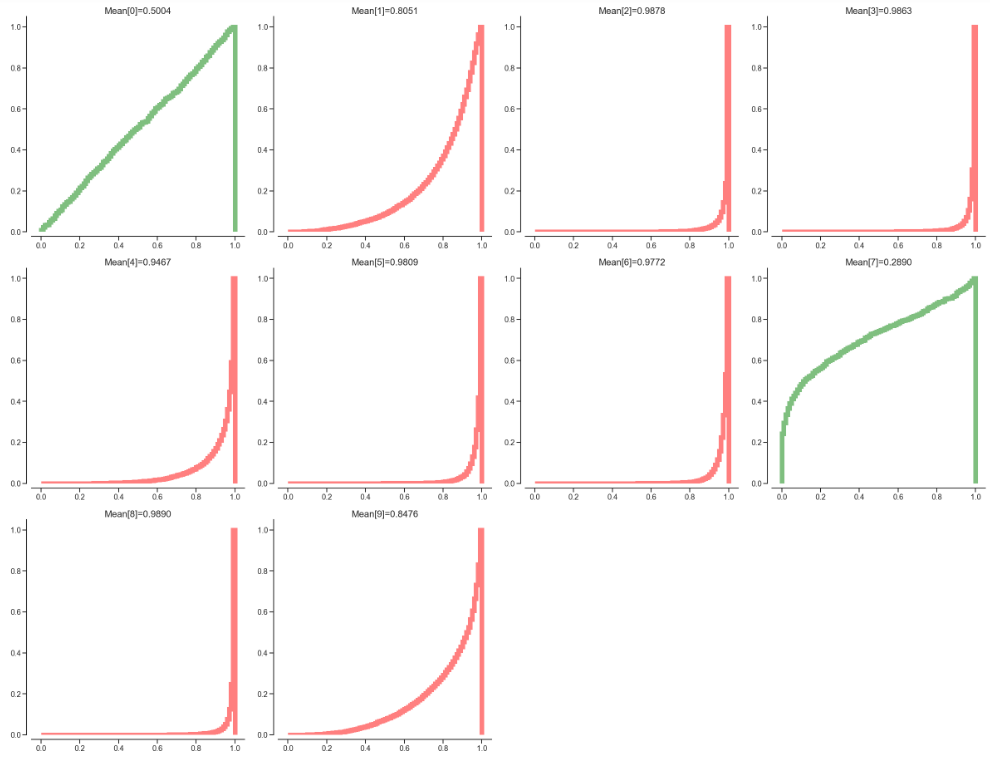


Fig 30 EVT plots showing CDF score vs reconstruction error for the known and unknown data.

Finally, we created a sample dataset from the test data. This sample dataset has 5000 samples. We then present the plots corresponding to the recall and accuracy for different values of alpha. We also label the recall and threshold and the corresponding accuracy for which the recall on both the known and unknown classes is the highest. Using this plot an appropriate value of threshold can be chosen for the model depending upon the use case.

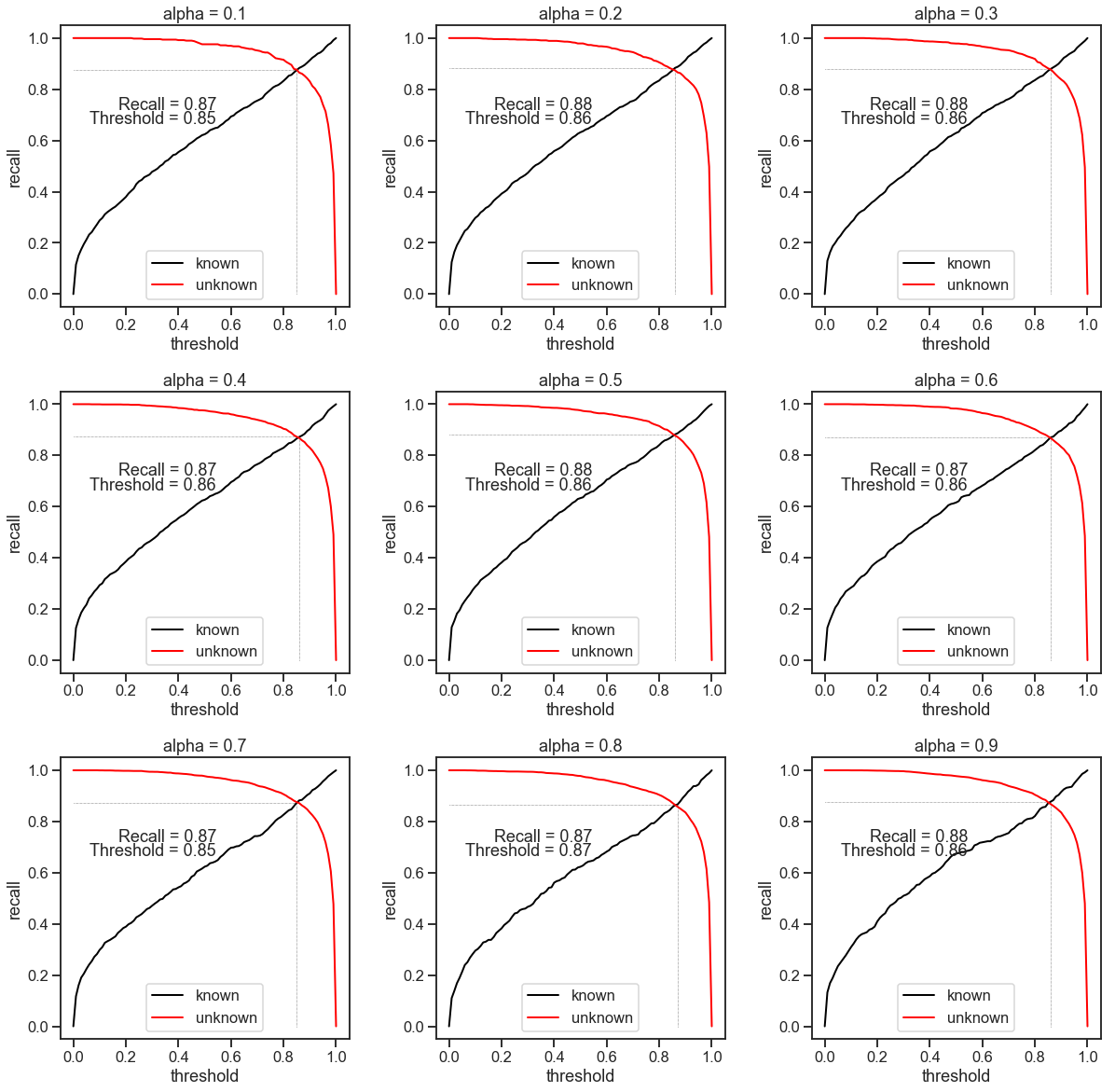


Fig31. Recall vs threshold plot for known digits ‘0’ and ‘7’

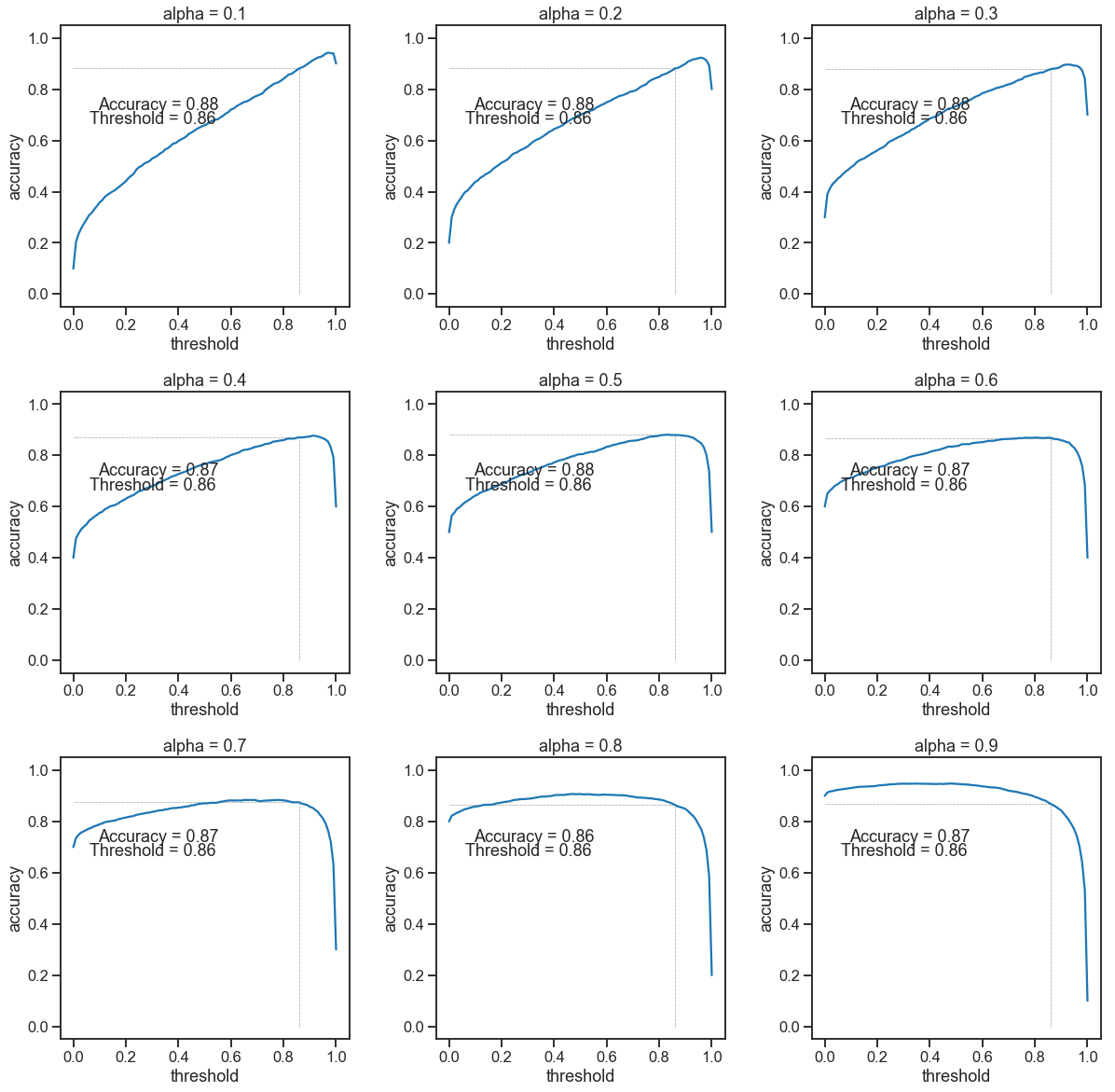


Fig32. Accuracy vs Threshold plots for known digits ‘0’ and ‘7’.

Figures 31 and 32 reveal that the accuracy depends on the no. of samples but the recall remains almost constant for different alpha. Thus, recall is a better metric for evaluation in this case. Furthermore, an alpha corresponding to 0.1 and 0.2 is much more realistic as unknown samples are quite rare. Thus, the plots for these alphas should be given more importance while deciding the value of the threshold.

4.3 Four Known Classes

Finally we applied our approach on multiple classes. As all possible combinations would be very large so we randomly selected four digits as the known class and trained the AAE using the data from those classes. We took the digits '0', '3', '7' and '9' as the known classes. The training loss curves are as follows.

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |
| Fig33. Adversarial Loss when '0', '3', '7' and '9' are known classes. | Fig34. Reconstruction Loss when '0', '3', '7' and '9' are known classes. |

|  |
| --- |
|  |

The reconstructed images of the training data are as follows.

|  |  |
| --- | --- |
|  |  |
| Fig 35 Original Images | Fig 36. Reconstructed Images |

|  |
| --- |
|  |

The images constructed by the decoder when sampling from the imposed prior distribution.

|  |
| --- |
|  |

Fig 37 Images from the imposed prior distribution.

Randomly sampling from the imposed prior distribution produces images that are from the known classes demonstrating that the regularization by the adversarial training was successful. During testing when the AAE was shown data from the unknown classes it reconstructed images that looked similar to the known classes and had a high reconstruction error on the unknown classes.

|  |
| --- |
|  |

Fig 38 Original Images of the unknown classes

|  |
| --- |
|  |

Fig 39 Reconstructed Images of the unknown classes

The histograms showing the reconstruction errors for all classes are as follows.

|  |
| --- |
|  |

Fig.40.Histograms showing the reconstruction errors per data for each class with the digits '0' , '3', '7'  and '9' as the known classes.

The following table contains the mean reconstruction errors obtained in all of the experiments.

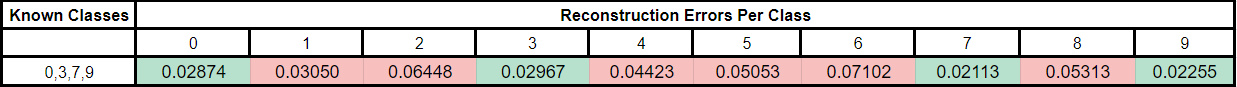


Fig 41. The reconstruction errors in 4 class experiments. The row contains the experiment with the columns having the mean reconstruction error for that class in that experiment.

The combined histogram of the reconstruction errors for the known and unknown data provides a better visualization of the results.

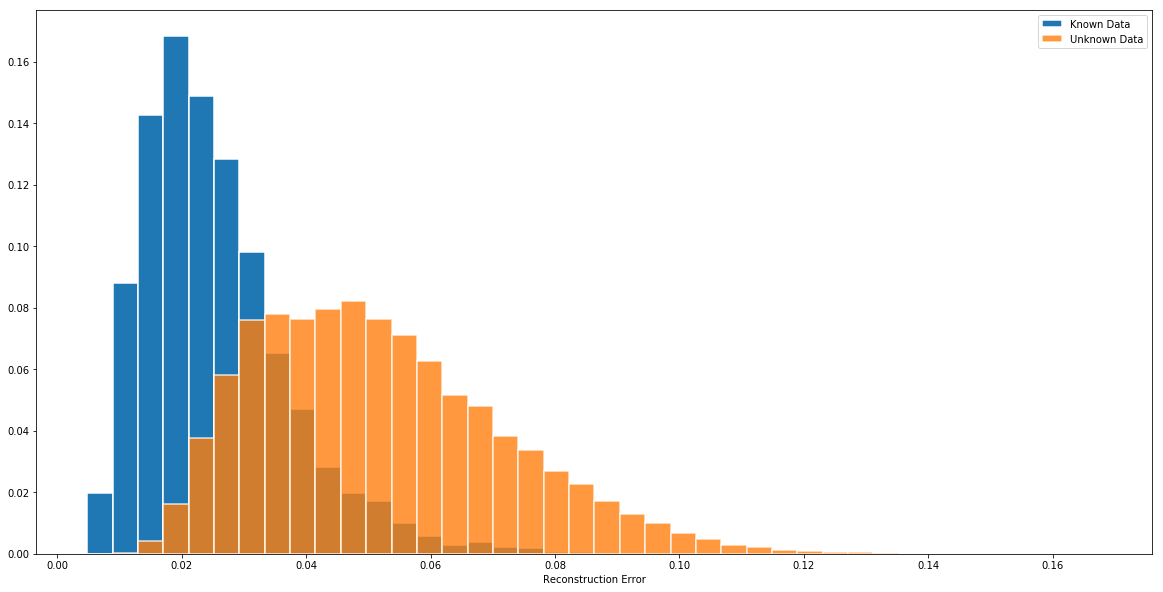


Fig 42. Combined Histogram plot. Blue plot is for known and Orange is for unknown data.

The histograms clearly show that there is a clear distinction between the reconstruction errors of the known and unknown classes with the errors being much smaller for the known classes. Finally the reconstruction errors of the known classes are fitted to an EVT distribution and the CDF scores for data belonging to the unknown classes is calculated.

|  |
| --- |
|  |

Fig 43 EVT plots showing CDF score vs reconstruction error for the known and unknown data.

The CDF score for most of the known classes is more than 90 percentile. The exceptional low score for the digit '1' is strange. One possible reason may be that the digit '1' is the easiest to reconstruct among all the numbers hence even though the AAE is not trained on '1' it is still able to get a sufficiently low reconstruction error for that class.

Finally, we created a sample dataset from the test data. This sample dataset has 5000 samples. We then present the plots corresponding to the recall and accuracy for different values of alpha. We also label the recall and threshold and the corresponding accuracy for which the recall on both the known and unknown classes is the highest. Using this plot an appropriate value of threshold can be chosen for the model depending upon the use case.

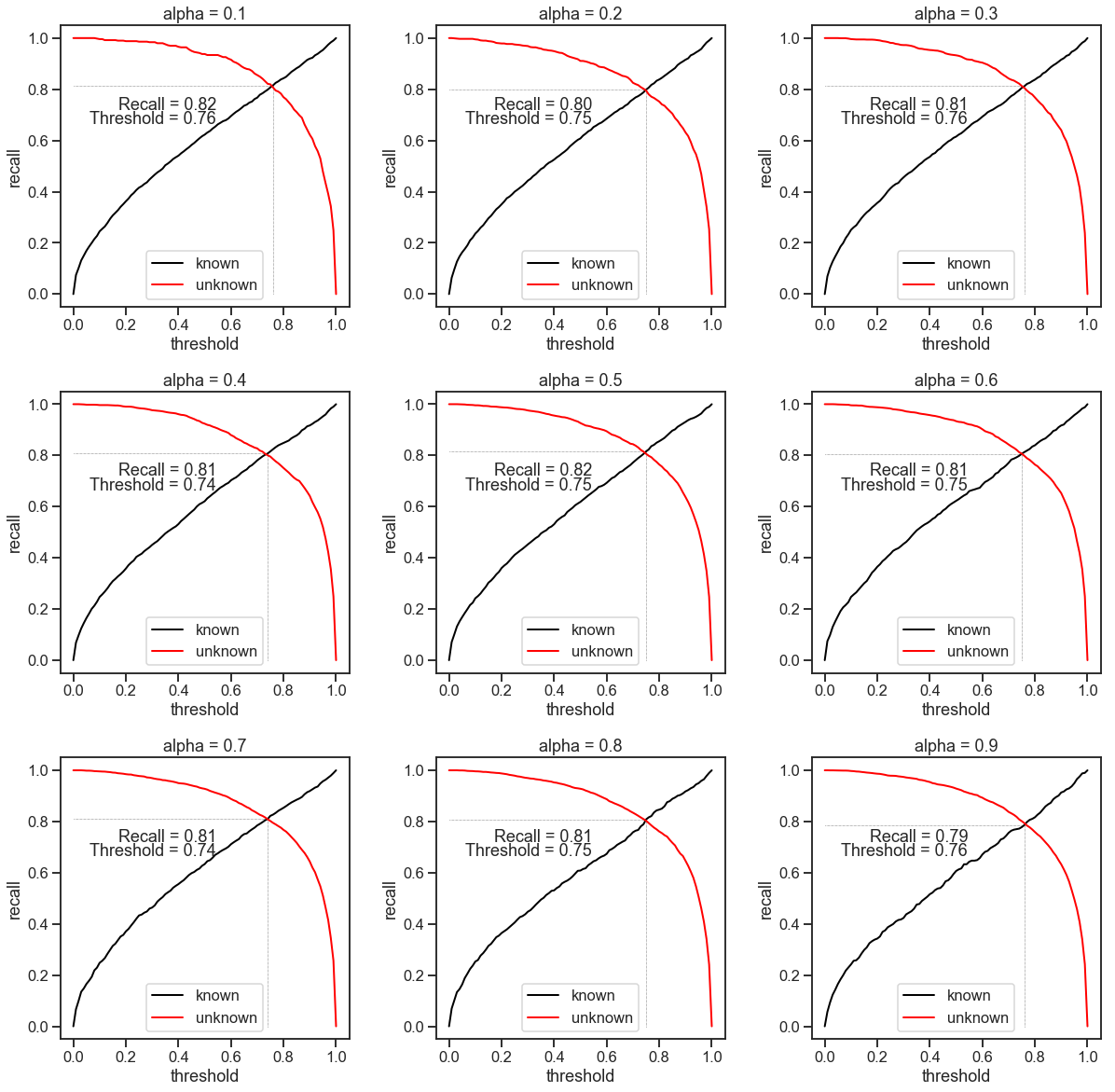


Fig44. Recall vs Threshold Plots where ‘0’ , ‘3’ ,’7’ and ‘9’ are known digits.

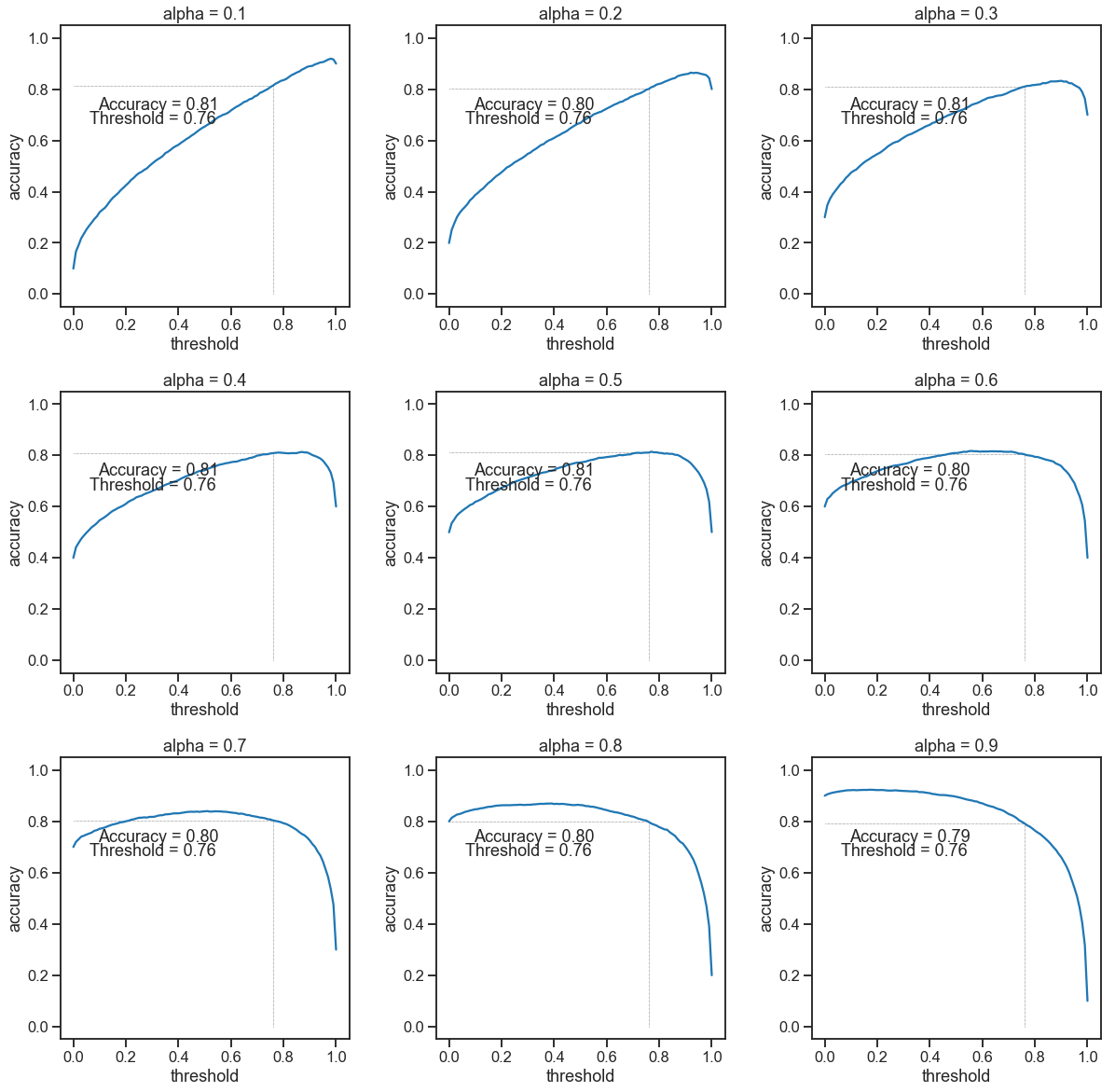


Fig45. Accuracy vs Threshold Plots where the digits ‘0’, ‘3’, ‘7’ and ‘9’ are known.

Figures 44 and 45 reveal that the accuracy depends on the no. of samples but the recall remains almost constant for different alpha. Thus, recall is a better metric for evaluation in this case. Furthermore, an alpha corresponding to 0.1 and 0.2 is much more realistic as unknown samples are quite rare. Thus, the plots for these alphas should be given more importance while deciding the value of the threshold.

4.4 DISCUSSION

The training of an AAE on some known data results in the AAE learning some inherent features of the known data. When it is given an unknown data the decoder still tries to reconstruct the input but as it only knows the features of the known classes, the reconstructed image looks just like the images from the known classes. This results in the AAE giving a low reconstruction score for the known classes while giving a much larger score for the unknown classes. In all of our experiments, we observed that the AAE was successful in getting a large reconstruction score for the unknown classes. This is clearly visible in the histograms showing the reconstruction errors. However our model does suffer from one fallback. As we can see in the reconstruction error tables and the error histograms that the reconstruction error for the digit '1' is very close to the reconstruction error of the known class, in some cases it is even less than the known class. A probable reason for this behavior is that the digit '1' is the easiest class to reconstruct and overlaps with many classes. Furthermore when we train the AAE using the digit '1' we find that the reconstruction score is exceptionally small as compared to other digits when they are used in training. Thus even though the reconstruction error for '1' is comparable with the known classes, it is actually much larger compared to the error when it was the known class. Thus we see that our model does try to increase the reconstruction error in all cases but the error for '1' is too small to begin with. So even after increasing the error it still remains comparable to the errors of the known class. Our model behaves in a similar way for classes that are quite similar to each other, eg. the digits '4' and '9'. A possible reason for this is the regularizing effect of the adversarial training. The adversarial training smoothens the output of the AAE for data on which it is not trained but belongs to the same prior distribution. This smoothening hence results in a low reconstruction error for the unknown classes which look similar to the known classes. We also see that the AAE finds the digit '8' the hardest to reconstruct. This is clearly observable from the reconstruction error table. This might be due to the model architecture. As the MNIST dataset is an image dataset, using an AAE that has convolutional layers would give better results on this dataset.

The combined histogram plots clearly show the error distinction in the known and unknown classes. This histogram can now be used to get a threshold that would be able to label the incoming data as known or unknown. The effectiveness of our approach is clearly visible in the EVT plots. We fit the reconstruction errors for the known classes to an extreme value distribution and then found the CDF score for the reconstruction error for the unknown data. In most cases, the CDF score was more than 90 percentile.

Deciding upon the evaluation metric for this problem was quite hard. We used recall as a metric as the dataset in this problem is highly skewed. It is also quite clear from the accuracy and recall plots that recall is pretty much invariant to the number of instances of each class in the dataset whereas the accuracy has a heavy dependence on alpha. Deciding on the threshold without the proper use case is a very difficult and an almost impossible task. We have presented the Recall vs threshold and Accuracy vs threshold plots, which give enough information to the user to decide the value of the threshold for his usage.

One thing that stands out in this study is that there is some dependence on the type of data used. For example, the digit ‘1’ has an exceptionally low reconstruction error irrespective of the class to which it belongs, whereas the digit ‘8’ has a significantly high reconstruction error in almost all of the experiments. While conducting the experiments taking different digits as known, we did not fine-tune the hyper-parameters each time and kept the same number of training epochs. This was done to remain consistent throughout the experiments. But it was observed that the training loss in some cases did not go as low as in some other cases. This anomaly for different classes behaving differently when taken as the known class may be because of this. In future work, it might be a good option to reach a consistent training loss and fine-tuning the experiments on the basis of the classes. This may generate consistent errors throughout each of the classes.

5 CONCLUSION AND RECOMMENDATIONS

We have seen that the use of regularized auto-encoders can be a good way to tackle the open world problem. Our approach of using an auto-encoder to generate a reconstruction error is similar to that used in [10]. However it reduces the time cost significantly for generating a reconstruction error for the open world samples. Further work can be done in this area by using different variants of the regularized auto-encoder like variational auto-encoder and the standard auto-encoder architecture. Furthermore work needs to be done in finding a thresholding technique that can be used for labelling the data into known and unknown classes based on the reconstruction error of the auto-encoder. Other versions of AAE's should also be tried like the unsupervised clustering AAE architecture. The use of generative models in generating data from the open space is also a promising approach towards solving the open world problem. Using the discriminator of the AAE is also a good research topic that may help in getting better results in this problem.

ACKNOWLEDGEMENTS

I am grateful to the Indian Academy of Science, Indian National Science Academy and the National Academy of Sciences, India for providing me the opportunity to work on this project. I express my sincere thanks to Professor Nikhil R. Pal, Indian Statistical Institute, Kolkata for his constant advice and guidance throughout my project which helped me get a good research experience this summer. I would also like to thank Mr. Biman Dey Sarkar, the staff of ISI Kolkata and my fellow lab mates who made my stay in Kolkata a comfortable and memorable one. I would also like to express my gratitude to Professor Anindya Chatterjee at IIT Kanpur, for providing my letter of recommendation to IAS, INSA and NASI. I would also like to thank my brother for helping me along the way. I am thankful to AuthorCafe for providing me a great platform to write my report.

References

1. Abhijit Bendale, Terence Boult, 2015 "Towards Open Set Discovery"
2. Zhiyuan Chen, Bing Liu, 2018 "Lifelong Machine Learning Ch.5"
3. Walter J. Scheirer et. al. 2014 "Probability models for open set recognition"
4. Bikram Karmakar, Nikhil R. Pal, 2017 "How to make a neural network say 'Don't Know' "
5. Debrup Chakraborty, Nikhil R. Pal, 2003 "A novel training scheme for multilayer perceptrons to realize proper generalization and incremental learning."
6. Ian Goodfellow, Yoshua Bengio, Aaron Courville "Deep Learning " Ch.14
7. Ian Goodfellow, et. al. 2014 "Generative Adversarial Networks"
8. Ian Goodfellow, NIPS 2016 Tutorial
9. Ian Goodfellow et. al. 2016 "Adversarial Autoencoders"
10. Huan-gang Wang, Xin Li, Tao ZHANG, 2016 "Generative adversarial network-based novelty detection using minimized reconstruction error"
11. Stuart Coles, "An Introduction to Statistical Modeling of Extreme Values".
12. Link to the scipy stats module: [docs.scipy.org/doc/scipy/reference/stats.html](https://docs.scipy.org/doc/scipy/reference/stats.html)
13. MNIST Dataset: [yann.lecun.com/exdb/mnist/](http://yann.lecun.com/exdb/mnist/)