technische universität
dortmund

Master Thesis

Improving Autoencoder Ensemble Learning for Unsupervised Anomaly Detection

Lecturers:
Prof. Dr. Emmanuel Müller
Simon Klüttermann

Author: Nikitha Rao

November 1, 2022

Contents

1 Introduction

Anomaly detection in data science has gained traction quickly due to its wide applications. The pursuit of solving mysteries is the essence of anomaly detection. As most datasets available do not have labels, the unsupervised anomaly detection mode is widely used. Unsupervised Anomaly Detection (AD) has many applications ranging from day-to-day activities like mobile fraud detection to critical industrial systems like network intrusion detection. Autoencoders are one of the unsupervised neural networks that are particularly helpful in learning interesting structures and identifying anomalies. Autoencoders are a simple three-layered network that takes input data and creates a compact latent space representation. When we try to recreate the input from this compact latent space, the data points that are hard to reconstruct are identified as anomalies. Thus autoencoder efficiently performs unsupervised anomaly detection through its unique structure. But building a single autoencoder model and hoping it would perform well is insufficient. Instead, we can create several models and leverage the power of combined results. This process is ensemble learning. Autoencoder combined with ensemble learning can be a very efficient anomaly detection tool as we merge two of the best machine learning concepts.

Autoencoders are robust neural networks. When we use several autoencoders in ensemble learning, they all tend to give similar results. But combining many similar autoencoders will not provide any significant results. We need diverse results to capture different input spaces and get a good result in ensemble learning. Techniques like Randnet (Jinghui Chen et al. 2017), where the connections of every autoencoder ensemble were different, have been used in the past. But the variability induced by this autoencoder ensemble was limited. Thus we proposed a novel algorithm called ASE (Autoencoder Synchronising Ensemble) in this thesis. ASE models consider the interactions between the autoencoder and increase the model variances.

The ASE algorithm is a novel algorithm that solves this problem of lack of variance in the results by considering the interaction between the autoencoder models of

ensemble learning. It concatenates the output of every autoencoder model. Network learning reduces losses that are an interaction between all the autoencoder models. The speciality of the ASE algorithm is its losses that consider the impact of one autoencoder model on the other. We try to mitigate several losses in ASE, and all these losses act on concatenated base model output. As we are considering the anomaly score after reducing the losses on the concatenated output, we do not have to worry about combining similar results and getting an ineffective ensemble. Additionally, considering the interactions between the base models give rise to a more meaningful result. We will conduct several experiments on 36 data sets and evaluate the performances of our models by checking the ROC scores. The results of our experiments show that ASE models can perform better than general ensemble learning models in most data sets if we fine-tune the parameters. All of our concepts, experiments and results are explained in the upcoming sections. Section 2 demonstrates a thorough analysis of anomaly detection methods using ensembles. Finally, our contributions to the ensemble-based anomaly detection and the related experiments and summary are elaborated in sections 3 to 6.

- Section 2 thoroughly explains the concept of anomaly detection, types of anomalies and solutions. In anomaly detection, four major types of anomalies, and two major kinds of solutions, are identified. This section also explains the concept of autoencoders and ensemble learning. This explanation builds a solid foundation for the anomaly detection methods proposed in this thesis. The last part of this section elaborates on the reason behind building a synchronising autoencoder ensemble.
- Section 3 proposes our novel ASE model for unsupervised anomaly detection. It also demonstrates several loss functions used in producing these anomaly detection models.
- Section 4 starts with introducing all the data sets used in our experiments. It then paints a picture of how the experiments progressed through the course of this thesis and the ASE model was built.

- Section 5 elaborates on all the results we got in the experiments. It compares the performance of the ASE model with the general autoencoder ensemble learning.
- Section 6 briefs the conclusions of our thesis and suggests some future work.

To summarise the main contents, this thesis researches the problem of anomaly detection, especially autoencoder ensemble learning, and advises several novel loss functions to promote better anomaly detection considering the interaction between various base models.

2 Existing concepts and the problem statement

2.1 Anomaly detection

People love solving mysteries. The pursuit of the unknown is thrilling and riveting, and this is essentially the essence of anomaly detection. Anomaly detection is a relatively new branch in data analysis, but it has quickly gained traction with numerous methods developed to detect anomalies. In layman's terms, anomaly detection is finding something unusual or discovering strange data behaviour. Mathematically, we can explain anomaly detection as follows. Consider a data set $X = x_1, x_2, x_3, ..,x_n$ where x_i belongs to R^d, n is the number of data points and d is the data dimension of this data set. Outlier detection is finding a subset x_{sub} that belongs to data set X but is inconsistent with other data points x_i that belong to X. A wide range of anomaly detection applications is from day-to-day activities like mobile fraud detection to critical industrial systems like network intrusion detection (Huang 2018). Earlier, anomaly detection systems were built manually by experts. But with increased volumes of data from various domains and the criticality of identifying anomalies on time, applying machine learning algorithms to detect anomalies automatically is important (Dunning and Friedman 2014).

2.1.1 Types of anomalies

There are several types of anomalies based on context and the volume of the anomaly points, and as shown in Figure 1, we can explain four types of anomalies:

- **Point Anomaly:** is a single data point that varies significantly from the rest of the data set. Consider data set $X = x_1, x_2, x_3, ..,x_n$. If this single data point x_i significantly varies from the rest of the data points in X, then x_i is a point anomaly. To understand in terms of a layman's example, consider an office with 100 employees. If 99 of the employees are present and 1 is on vacation, then that 1 person can be a point anomaly.

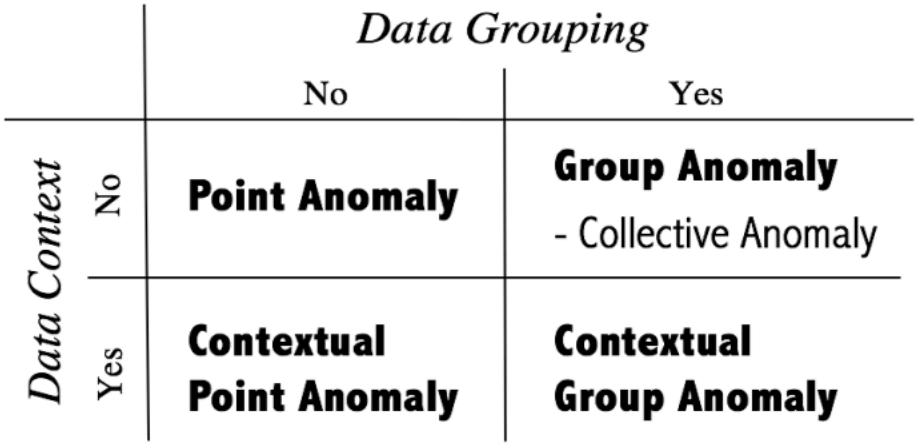

Figure 1: Different types of anomalies (Huang 2018)

- **Group Anomaly:** is a set of observations that varies significantly from the rest of the data set. Consider data set $X = x_1, x_2, x_3, ..,x_n$ where x_{sub} is a subset of data points that belong to X. If this subset of data points x_{sub} significantly varies from the rest of the data points in X, then the group of data points belonging to x_{sub} is a group anomaly. To understand this in terms of a layman's example, again consider an office with 100 employees. If 85 of the employees have a university degree and 15 employees do not have any degrees, then these 15 employees can be considered as a group anomaly in a company with the majority of university graduate employees.
- **Contextual Point Anomaly:** is a data point that behaves like a point anomaly only under certain pre-conditions. To understand this in terms of a layman's example, again consider the example of office. If 1 person visits the office, it is expected behaviour on weekdays, but this person visiting the office seems to be an anomaly on weekends.
- **Contextual Group Anomaly:** is a set of observations that behaves like a Group anomaly only under certain pre-conditions.

2.1.2 Modes of anomaly detection

We build a model for anomaly detection by training it on the available data sets. These data sets may be labelled to indicate normal and abnormal data. Collecting these well-defined data sets with labels that accurately represent real-world data sets is tedious. Experts in the respective fields must do the labelling, which is costly and time-consuming. Based on the availability of the labels of the data sets, we can broadly have two modes of anomaly detection. There can be more modes based on the nitty-gritty of the availability of data labels. But for a simple understanding, we will consider only two modes of classification (Ibidunmoye, Hernández-Rodriguez, and Elmroth 2015):

- Supervised anomaly detection: The data sets need to be labelled in supervised anomaly detection. Each point in the data used to train either belongs to the normal or anomaly class. The aim is to build a generic model that will capture the relationship between the feature values of the data sets and decide if the new data sets are normal or abnormal. As this learning heavily depends on the availability of accurately labelled data sets, it limits its real-time application. It is hard to collect labelled data sets for every problem statement. Thus they do not apply to identify unknown anomalies as well. But due to the high accuracy of anomaly identification, they are helpful in scenarios where the anomalies are already known (Gavrilova 2021).
- Unsupervised anomaly detection: Most frequent problem in training the models is the unavailability of labelled data sets. In this frequently occurring scenario, we can use unsupervised anomaly detection. Unsupervised learning aims to learn the irregularities and hidden patterns in the data set. The data is classified into normal and anomalous based on statistical properties like the density of the distribution. Normal instances are assumed to be more than anomalous instances. The main advantage is that the high cost of labelling the data is saved, and unknown anomalies can be identified. The disadvantage is that known anomalies can be missed, as the accuracy can be lower than the supervised anomaly detection (Gavrilova 2021).

2.2 Autoencoders

To perform any machine learning tasks like the classification of images or fraud detection in credit card transactions, we need a large volume of data. We need image labels, fraud transaction labels, or other labelled data to perform these diverse machine-learning tasks. As discussed earlier, it is hard to get the labelled data as it needs expert input. We want to leverage the data that is already available without going into the hassle of labelling data. Autoencoders are helpful in this regard. Autoencoders are unsupervised neural networks that help in learning interesting things like the structure of the unlabelled data that helps make several decisions related to the data (Dertat 2017).

2.2.1 Overview of the autoencoder structure

Autoencoder networks perform two main actions. Firstly, the network compresses the given input data into a compact representation. A latent space is a lower dimensional representation of the input data. The second action is reconstructing the input data from this lower dimensional latent space. The reconstruction error is the difference between the reconstructed and the original input. The network is trained to reduce this reconstruction error, and hence the network learns to exploit the actual structure of the data to create the efficient lower dimensional latent space. An overview of the structure can be found in Figure 2. The structure has three main components

- Encoder: The left side of Figure 2 highlighted in yellow is the encoder. It is a fully connected feed-forward network which accepts the input x_i. The real data has some structure, and we do not need every part of the full input space to represent data. Using the encoder, we can map the input from full input space to a lower dimensional representation. The encoder approximates a function that maps the data from the full input space into a lower dimensional system that takes advantage of the structure of the data.

It is a layer of a neural network that is fully connected where we can decide the number of layers and neurons.

- Latent space: The middle layer of the Figure 2 represented as $h(x_i)$ is the latent space. It is a single layer of neurons that has the most compact representation of the input data. We can decide the number of neurons in this single layer.
- Decoder: The last layer of Figure 2 highlighted in green is the decoder. The encoder's output is used by the decoder to attempt to recreate the original input. In short, it tries to reverse the encoding process. The layers and neurons in the decoder are commonly similar to that of the encoder.

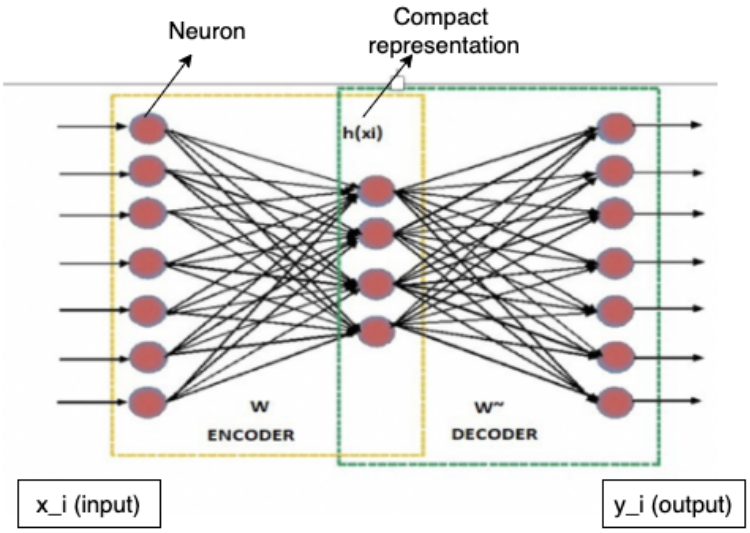

Figure 2: A network diagram of an autoencoder with 3 layers: encoder, latent space and decoder.

2.2.2 Parameters of autoencoders

Several factors need to be decided before building the autoencoders. These parameters are essential to determine the performance of an autoencoder. Below are the parameters that must be considered (Bandyopadhyay 2022).

- **Activation function:** An essential component of a neural network's architecture is its activation functions. How successfully the network model learns the training dataset will depend on the activation function used for the hidden layer. The weighted total of the input is turned into output in a network layer according to an activation function. Our thesis uses different activation functions for encoders and decoders after careful iterations (Brownlee 2021).
- **Layers:** The layers in the encoder and the decoder is a crucial hyperparameter for tuning autoencoders. A smaller depth is quicker to process, whereas a large number of layers increases model complexity.
- **Structure parameter (α):** The structure parameter α controls the number of nodes in different layers. The ratio of nodes from the previous layer that should be in the current layer is α. Larger α values indicate more nodes in the network. Therefore it seems like they would have more potent modelling capabilities. However, higher modelling ability does not automatically equate to greater accuracy because neural networks tend to overfit (Jinghui Chen et al. 2017).
- **Reconstruction Loss:** The type of input and output we want the autoencoder to adapt to significantly impacts the loss function we employ to train it. For instance, MSE Loss and L1 Loss are the most often used loss functions for reconstruction in image data. We employed several new losses in our thesis and examined how each performed.

2.2.3 Anomaly detection in autoencoders

In autoencoders, we train the network on the normal data set. The encoder reduces this normal data set into a lower dimensional latent space. Then, the decoder tries to reconstruct the normal data set. The difference between the predicted and original inputs is the reconstruction error or anomaly score. The network is trained to reduce the reconstruction error and thus improve the reconstruction of the decoder. When this network is exposed to anomalous data, the

reconstruction error is higher as the network was trained on normal data. Therefore, the reconstruction error or anomaly score will be higher in anomalous data points. When the anomaly scores exceed a certain threshold, the data point is considered to be an anomaly.

2.3 Ensemble learning

The goal of a machine learning algorithm is to train a model efficiently to solve the intended problem, and instead of building a single model and hoping that it performs well, we can create several models and leverage the power of combined results. This is the basis for the concept of ensemble learning.

2.3.1 Overview and types of ensemble learning

Certain models may perform excellently in one aspect of the data, while other models might capture other data parts. Instead of learning one model and losing certain aspects of learning from the data, we can model several simple models and combine their outputs to produce the final decision. In ensemble learning, the combined strength of the models rectifies the single model variance and bias. Ensemble learning provides results where the final accuracy is better than the individual results. There are three main ensemble learning types depending on these algorithms' purpose. They are (CorporateFinance 2021, Otieno 2018):

- **Bagging:** Combining aggregation and bootstrapping is bagging (CorporateFinance 2021). In bootstrapping, the replacement technique creates samples from the whole population. The sampling with replacement approach aids in the randomization of the selection process. All potential outcomes of the prediction are included in the aggregate, which then randomizes the result. As a result, the aggregate is based on every result of the prediction models. This approach has the benefit of combining weak base learners into a more stable single strong learner. However, bagging is computationally costly.

- **Boosting:** Boosting is an ensemble strategy that improves future predictions by learning from prior predictor errors. Boosting falls within the sequential ensemble learning category that we discuss next. Boosting works by placing weak learners in sequential order so that they can learn from the subsequent learner to improve their prediction models. Gradient boosting, Adaptive Boosting (AdaBoost), and XGBoost (Extreme Gradient Boosting) are a few types of boosting algorithms.
- **Stacking:** The stacking approach allows a training algorithm to combine the predictions of multiple different learning algorithms that are comparable. Stacking aims to investigate the space of different models for the same problem, in contrast to how we approach a learning issue with numerous models that can learn specific aspects of the problem but not the entire problem space. As a result, we may create several unique learners, employ them to make an intermediate prediction, and then incorporate a new model that gains knowledge from the intermediate predictions of the same target. The application of stacking in regression, density estimations, distance learning, etc., are only a few examples.

There are two categories of ensemble learning for anomaly detection depending on how the models are built. They are (CorporateFinance 2021, Otieno 2018):

- **Sequential ensemble learning:** This method sequentially creates base learners. It makes use of the interdependence between the base learners. The outcomes of one base learner complement those of other base learners, just like a company's corporate team addressing an issue and coming up with a solution by combining all of their skills. When a base learner makes a prediction, another base learner uses it and attempts to make the following prediction with fewer errors. After that, a third base learner uses the previously suggested prediction and makes yet another attempt to bring the error down. This process continues until the prediction is pretty accurate. An example is the AdaBoost algorithm (Schapire 2013) in supervised learning.

- **Parallel ensemble learning:** The independence between the base learners is exploited in this technique, which generates base learners in a parallel style. In a few parallel ensembles, every base learner predicts an output, and the prediction with the highest likelihood is chosen. In a few other parallel ensembles, the base learners pick up knowledge from various data set samples and combine their knowledge to produce predictions. An example is the random forest algorithm.

2.3.2 Using autoencoder with ensemble learning

From Section 2.2, we know that autoencoders are unsupervised neural networks that reconstruct the output similar to the input. From Section 2.3, we know that ensemble learning combines the results from several base models to get optimal output. We can bring the best of both worlds in anomaly detection by integrating the unsupervised anomaly detection of autoencoders with optimally combined outputs of ensembles.

In autoencoder ensembles, the base models are autoencoders. Every autoencoder base model tries to reconstruct the output from the given input. It scores each data point with a score that is a reconstruction error. Every data point has a different score from every autoencoder base model. All of these scores are combined through one of the ensemble learning techniques to get an optimal score for each data point. The data point is identified as an anomaly if this combined score of a data point is above a certain threshold. This threshold can be manually decided or calculated by various techniques (Sinch_blog 2021).

2.4 Improving autoencoder ensemble learning

When we use ensemble learning for anomaly detection, we have to combine the anomaly score from every base model. But this is ineffective if we have similar base model results. Of course, one can use sampling or any other ensemble technique to capture different spaces and vary the predictions. But the variability provided

in these cases is limited. Autoencoders give similar results when we use them as base models. Therefore, there is less variability and high correlation in learning (Klüttermann and Müller 2022).

Further, when the data available is limited, autoencoders tend to overfit. Using ensemble learning may limit the tendency of autoencoders to overfit. But these do not guarantee avoiding overfitting. Techniques like Randnet (Jinghui Chen et al. 2017) have previously been used for autoencoder ensembles. In this method, the connections in autoencoders are randomly changed for every base model. However, the variability they induced was limited.

Additionally, every base model learns independently in autoencoder ensembles. But considering the interactions between these autoencoder base models may be helpful as they all try to achieve the same purpose of identifying anomalous data points. Thus we have developed a novel algorithm ASE (Autoencoder Synchronising Ensemble), explained in Section 3.1. ASE considers the interactions between the autoencoder and increases the variance of the models.

2.4.1 Evaluating criteria: ROC

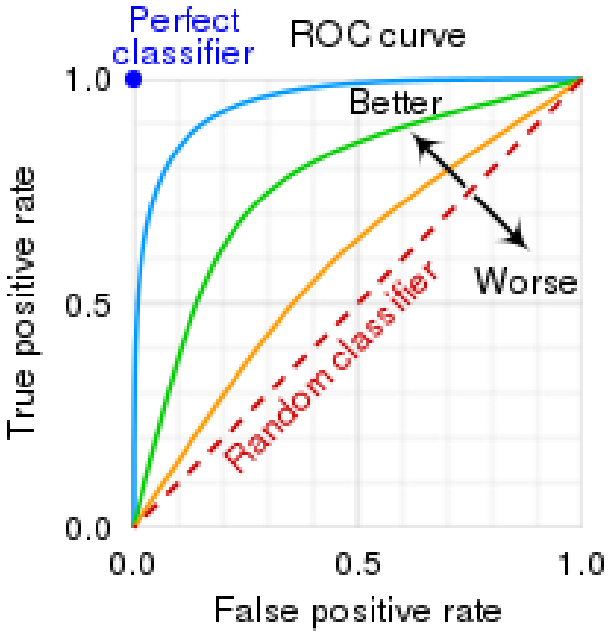

Figure 3: A diagram representing the ROC plot with FPR versus TPR. (Wikipedia 2022b)

We use ROC-AUC scores to evaluate the performance of our ASE algorithm and compare them with other algorithms. It is a commonly used success metric. ROC curves show the ratio of true positives to false positives. TPR (true positive rate) is plotted against FPR (false positive rate) on the ROC curve, with FPR on the x-axis and TPR on the y-axis. A measure called area under the curve (AUC) is the area under the resulting monotone curves. AUC enables the comparison of the findings quantitatively and the display of several outcomes in a single graph. The area that results from a random result will roughly occupy half of the available space because both the real positive rate and the false positive rate will increase simultaneously. Figure 3 illustrates how the area under the curve will perfectly fill the allotted space for an ideal result, returning all outliers and the inliers only after that. Therefore, the maximum value is 1.0. The likelihood that two randomly selected objects, one of which is an outlier (positive example) and the other are an

inlier (negative example), would be correctly sorted may be characterized as the ROC AUC value. ROC curves and ROC-AUC analyses are particularly well-liked for evaluating outlier identification because they naturally solve the problem of imbalance in the classes by using relative frequencies (Hanley and Mcneil 1982, Zimek, Campello, and Sander 2014, Huang 2018).

3 Our contributions and methods

This section highlights how the ensembles are implemented in this research and elaborates more on different loss functions used to optimize the result.

3.1 ASE (Autoencoder Synchronising ensemble) model

The widely used ensemble method is parallel ensemble method that we discussed in Section 2.3.1. In the parallel ensemble learning method, we build every base model independently. The output of one base model does not affect the other. Moreover, autoencoders are used as base models in unsupervised anomaly detection. In autoencoders, the encoding layer accepts the input, the latent space layer stores the compact representation of input space, and the decoder layer tries to reconstruct the output from the hidden layer. The detection of the anomalies depends on the reconstruction error during the decoder reconstruction. When we apply ensembles onto autoencoders, the loss function we use is just an MSE loss (3.2) over the features. The result we get from each base model after MSE loss optimization is then combined to get a single output. As explained in Section 2.4, it suffers from the lack of variation and high correlation and does not realize the full capabilities of ensembles (Klüttermann and Müller 2022).

The ASE algorithm is a novel algorithm that solves this problem of high correlation and low variance in the results by considering the interaction between the base models. The issue of combining non-diverse results is solved as ASE concatenates the output of every base model. The backpropagation reduces losses that are an interaction between all the base models. We try to mitigate several losses in ASE, and all these losses act on concatenated base model output. As we are considering the anomaly score after reducing the losses on the concatenated output, we can handle combining similar results and getting an ineffective ensemble. Correlation in ASE is also reduced by the way the losses are formulated, which can be seen in the next section. Additionally, considering the interactions between the base models give rise to a more meaningful result. We tried to understand the working

of an ensemble model by seeing a corporate company example in Section **??**. If we consider a similar example, consider a corporate team with five members, and all five members are working to build the same project. Continuous interaction between these five team members during the project will lead to more meaningful results than all five working independently and then combining their results. Similar is the impact of interactions among the base ensemble in our ASE model. As we are building all base models synchronously, the model's name is autoencoder synchronizing ensemble (ASE).

Figure 4 is the pictorial representation of our network. The input is taken by the input layer and fed into each network in the figure. These networks are typical autoencoders with encoder, latent space and decoder layers. Each encoder layer that accepts input creates a compact representation, and each decoder layer tries to reconstruct the inputs. The outputs of all the decoders are concatenated. Several losses optimize this concatenated output. The anomaly score of each data point is calculated during this optimization process. The data point is marked as an anomaly if this score is above a certain threshold (Sinch_blog 2021).

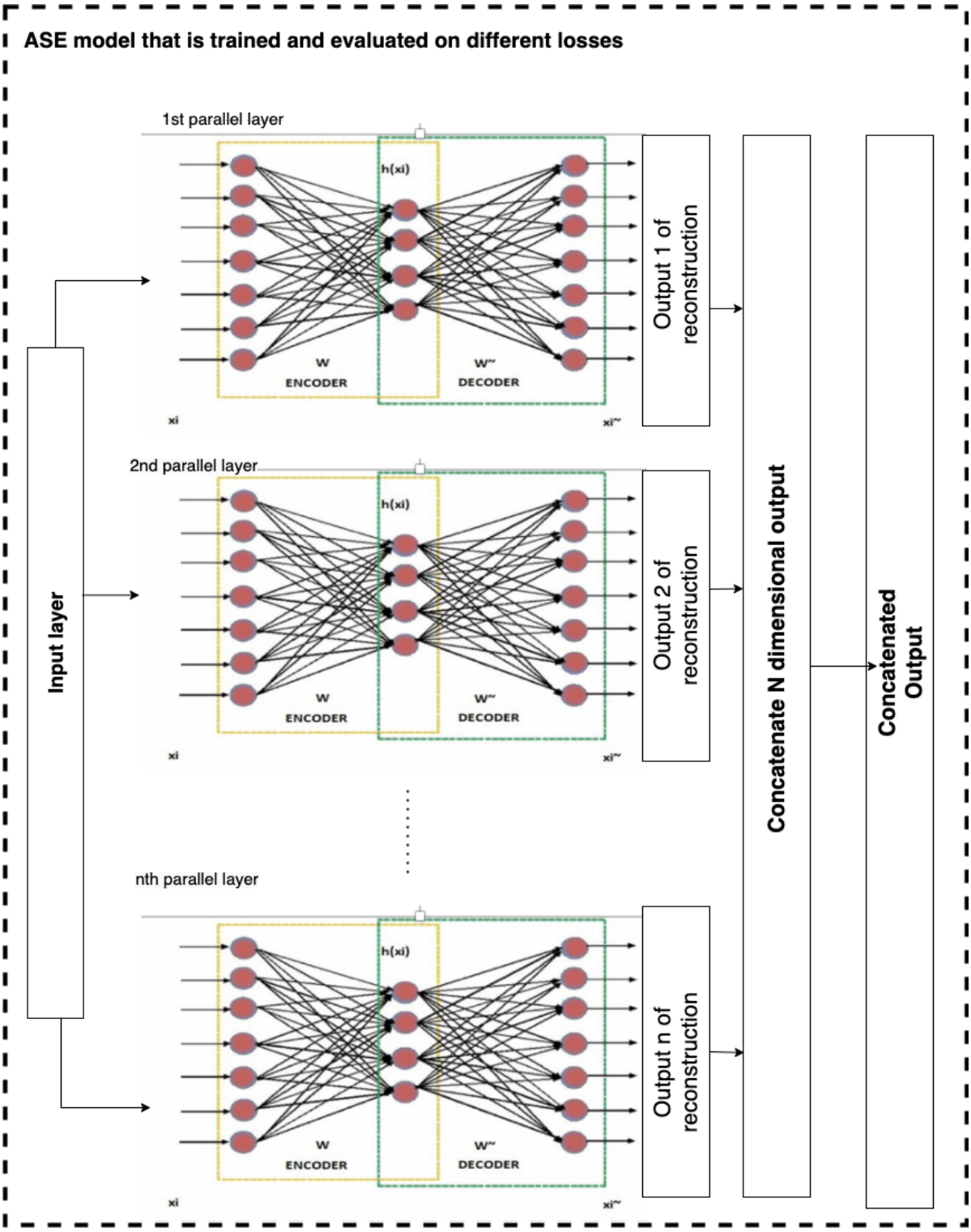

Figure 4: A network diagram of our synchronising auto encoder ensemble

3.2 Mean squared error loss

$$\text{MSE} = \frac{1}{d}\sum_{i=1}^{d}(x_i - f(x)_i)^2 \tag{1}$$

The mean squared error or MSE gauges the degree of inaccuracy in statistical models. Between the observed value x_i and projected values $f(x)_i$, it evaluates the average squared difference of every feature d. The MSE is equal to 0 when a model is error-free. Its value grows when the model error does as well. The mean squared deviation is another name for the mean squared error (MSD) Frost 2019. The MSE loss is the most used loss to train the general autoencoder ensemble model.

3.3 Multi dimensional MSE loss

$$\frac{1}{N}\sum_{m=1}^{N}\frac{1}{d}\sum_{i=1}^{d}(x_i - f^m(x)_i)^2 \tag{2}$$

This loss is similar to the loss we optimise in general ensemble models. But as we are concatenating outputs of several synchronising ensembles, it has an additional step to average the squared difference across each synchronising autoencoder base model. The squared difference between the original input x_i and the model output $f^m(x)_i$ is averaged across the data features d. This is the exact calculation made in general ensemble learning. An additional step is that the squared difference between the original input x_i and the model output $f^m(x)_i$ is also averaged across the number of synchronous ensembles N. This error is then optimised to get a good model. Hence if one synchronous ensemble gives a poor result, our model compensates for the mistake by considering the average of all the synchronous models. This loss is called loss_1 in our further explanations and experiments.

3.3.1 Modifications to Multi dimensional MSE loss

The loss_1 explained in previous section is the same as general anomaly ensemble learning loss. To consider the interactions between several models, we slightly modify the loss_1. Below are some of the modifications to loss_1.

Max Multi dimensional MSE loss

$$\max_{N} \frac{1}{d} \sum_{i=1}^{d} (x_i - f^m(x)_i)^2 \tag{3}$$

This loss is a modification to the multi-dimensional MSE loss. In the multi-dimensional MSE loss, we calculated the average MSE across N synchronizing ensembles. But in this loss, we select the synchronizing ensemble, which gives a maximum MSE loss across all the features. The loss prioritizes the model whose output differs hugely from the actual output. Hence we try to optimize the worst of N synchronizing models. As a result, all the predictions are optimized, and the model has more freedom as we select only one synchronizing base autoencoder. This loss is called max_loss_1 in our further explanations and experiments.

Min Multi dimensional MSE loss

$$\min_{N} \frac{1}{d} \sum_{i=1}^{d} (x_i - f^m(x)_i)^2 \tag{4}$$

In this modification to the multi-dimensional MSE loss, instead of calculating the average of MSE across N synchronizing ensembles, we select the synchronizing ensemble that gives a minimum MSE loss across all the features. The loss prioritizes the model whose output is similar to the actual output. Hence we try to optimize the best of N synchronizing models. The training task is now simplified as we need one autoencoder base model to perform well for every data sample. This loss is called min_loss_1 in our further explanations and experiments.

Median Multi dimensional MSE loss

$$\text{med}_N(\frac{1}{d}\sum_{i=1}^{d}(x_i - f^m(x)_i)^2) \tag{5}$$

The strictness of the worst model and the best model selection is loosened in this loss. Taking the median instead of the maximum and minimum may combine the best of these losses. This loss is called median_loss_1 in our further explanations and experiments. A model that gives a median loss among all the MSE losses across features for N synchronizing ensembles is selected in loss optimization of median_loss_1.

3.4 Ensemble Averaged MSE loss

$$\frac{1}{d}\sum_{i=1}^{d}(x_i - \frac{1}{N}\sum_{m=1}^{N} f^m(x)_i)^2 \tag{6}$$

In the loss_1, we calculate the MSE across all features, aggregate it and then aggregate the MSE across all synchronising ensembles. Thus, this behaved similarly to a typical ensemble loss. But in this loss, the first step is to aggregate the predictions of all the synchronising ensembles. The difference between the mean prediction from all the synchronising ensembles and the actual input is then calculated for each feature. Finally, the square of this error is averaged. In short, the average of predictions $f^m(x)_i$ from all the N synchronising ensembles is used to calculate the mean squared error from actual value x_i across features d. Hence, the output from every ensemble matters as we average all of the synchronising ensemble outputs at first. As many models pitch in to predict each feature, we can learn complicated structures. This loss is called loss_2 in our further explanations and experiments.

3.4.1 Modifications to Ensemble Averaged MSE loss

The loss_2 explained in the previous section weighs every output of the synchronising ensemble equally as we take an average. But if one of the synchronising

ensembles performs badly. This might lead to some bad results. Thus we slightly modify the losses in the upcoming sub-sections.

Max Ensemble Averaged MSE loss

$$\frac{1}{d}\sum_{i=1}^{d}(x_i - \max_{N}\sum_{m=1}^{N} f^m(x)_i)^2 \tag{7}$$

This loss is a modification to the ensemble-averaged MSE loss. In the loss_2, we calculated the average of MSE across N synchronising ensembles before we compared it to the original input x_i. But in this loss, the synchronising ensemble with the highest prediction value is considered and compared with each feature's initial input. This loss is called max_loss_2 in our further explanations and experiments.

Min Ensemble Averaged MSE loss

$$\frac{1}{d}\sum_{i=1}^{d}(x_i - \min_{N}\sum_{m=1}^{N} f^m(x)_i)^2 \tag{8}$$

In this modification to the ensemble-averaged MSE loss, instead of calculating the average of MSE across N synchronising ensembles, we select the synchronising ensemble that gives the lowest prediction value for each feature. This loss is called min_loss_2 in our further explanations and experiments.

Median Ensemble Averaged MSE loss

$$\frac{1}{d}\sum_{i=1}^{d}(x_i - \mathrm{med}_{N}\sum_{m=1}^{N} f^m(x)_i)^2 \tag{9}$$

We are considering the median prediction value of the predicted output for each feature in N synchronising ensembles. This is likely helpful as it is not susceptible to extreme values like a mean. This loss is called median_loss_2 in our further explanations and experiments.

3.5 Cross combination multiplier loss

$$\frac{1}{N}\sum_{ma=1}^{N}\frac{1}{N-1}\sum_{mb=ma+1}^{N}\left|\sum_{i=1}^{d}(x_i - f^{ma}(x)_i)\cdot\left(x_i - f^{mb}(x)_i\right)\right| \tag{10}$$

Cross combination multiplier loss is called loss_3 in all the further explanations and experiments. The loss_3 is a unique loss that is newly formulated with an intuition of how a covariance (Moody 2019) might work. If we look past the complexity of how the formula is written, $(x_i - f^{ma}(x)_i)\cdot\left(x_i - f^{mb}(x)_i\right)$ is essentially just checking how the output $f^{ma}(x)_i$ from one synchronizing ensemble ma is varying with another output $f^{mb}(x)_i$ from another synchronizing ensemble mb, across all the features. In short, we try to optimize a function similar to the covariance. Optimizing this function minimizes the loss in the model. The absolute we calculate ensures no information loss when we have varying negative and positive outputs from every synchronizing ensemble model.

3.6 Cross combination MSE multiplier loss

$$\frac{2}{(N)(N-1)}\sum_{ma=1}^{N}\sum_{mb=ma+1}^{N}|\frac{1}{X}\cdot\sum_{i=0}^{X}(\frac{1}{d}\sum_{j=0}^{d}(f^{ma}(x_i)^j - x_i^j)^2)\cdot(\frac{1}{d}\sum_{j=0}^{d}(f^{mb}(x_i)^j - x_i^j)^2)| \tag{11}$$

Cross-combination MSE multiplier loss is called loss_4 in all the further explanations and experiments. We know that loss_3 was a new loss based on the intuition of a covariance. The loss_4 is written to overcome some of the shortcomings of the loss_3. In loss_3, if one model is a perfect prediction for one feature, we cannot optimise the rest of the models for that feature. This is because the formula is, $(x_i - f^{ma}(x)_i)\cdot\left(x_i - f^{mb}(x)_i\right)$. If mb ensemble has a perfect prediction for a particular feature d, $x_i - f^{mb}(x)_i$ becomes 0. When we multiply any other value with 0, we get a 0. Hence we square this difference $x_i - f^{mb}(x)_i$ and average it across all the features d before we multiply it with the difference from another model. We also average the entire result across the total data points X to overcome any loss of information that might occur in loss_3. Finally, we averaged it across all the ensembles to find a mean squared error. In short, loss_4 compares

the squared error across all the features of one ensemble model with the other for every data point. Hence if we want a low loss, we need a good reconstruction.

4 Experimental setup

This section begins with an overview of the data sets that we used. The step-by-step progression of our model throughout the thesis experiment and discussion of the results we obtained in each step is discussed next.

All the experiments explained below are written in the programming language *Python*3.0. It is an object-oriented, high-level programming language (Van Rossum and Drake 2009). The packages that we used are: TensorFlow for building neural networks (Abadi et al. 2016), matplotlib for visualising the data (Hunter 2007), pandas for data structures (McKinney et al. 2010), seaborn for more refined data visualisation (Waskom 2021), scikit for statistical modelling (Pedregosa et al. 2011), tqdm for seeing the code progress (Costa-Luis 2019), scipy for scientific computations (Virtanen et al. 2020), and finally NumPy for mathematical calculations (Harris et al. 2020).

4.1 Overview of Data sets

There are 36 data sets used in our experiment to train the model. The details of each data set, what it signifies, and its anomalies can be found in Appendix 6. Below is a table that gives a bird's eye view of the attributes of each data set:

Table 1: This table represents the training data size and the number of anomalies in each of the above data sets.

Data Set Name	**Shape of training data**	**Number of anomalies**
Arrythmia	[320, 257]	66
Cardio	[1479, 21]	176
Concardia3_32	[280, 253]	120
Delftpump_5x3_noisy	[152, 64]	64
Delftpump_AR	[133, 160]	56
Elevators	[8029, 18]	3440
Gas drift	[1796, 128]	769

Table 1: This table represents the training data size and the number of anomalies in each of the above data sets.

Data Set Name	**Shape of training data**	**Number of anomalies**
HeartC	[96, 13]	41
Hepatitis	[91, 19]	32
Hill valley	[425, 100]	181
Housing low	[410, 13]	48
Imports	[50, 24]	21
Ionosphere	[158, 33]	67
Letter	[1400, 32]	100
Magic telescope	[8633, 11]	3699
Mnist	[6203, 78]	700
Musk	[2868, 166]	97
Opt digits	[4916, 62]	150
Ozone level 8hr	[112, 72]	48
PC3	[1243, 37]	160
Pen digits	[6558, 16]	156
Qsar-biodeg	[250, 41]	106
Satellite	[3080, 36]	1319
Segment	[231, 18]	99
Sonar mines	[78, 60]	33
Spam base	[1952, 57]	836
Spectf_0	[67, 44]	28
Speech	[3564, 400]	61
Steel plates fault	[282, 27]	120
Vehicle van	[140, 18]	59
Vowels	[1356, 12]	50
Waveform5000	[1185, 40]	507
Wbc	[336, 30]	21
Wbc non ret	[106, 33]	45
Synthetic linear data set	[10000, 2]	2000

Table 1: This table represents the training data size and the number of anomalies in each of the above data sets.

Data Set Name	Shape of training data	Number of anomalies
Synthetic donut data set	[10000, 2]	2000

4.2 Initial model for autoencoder ensemble learning

As a first step, we built an existing sequential autoencoder ensemble learning model for three of the data sets: Cardio, Gas drift, and Satellite.

4.2.1 Overview of the autoencoder used

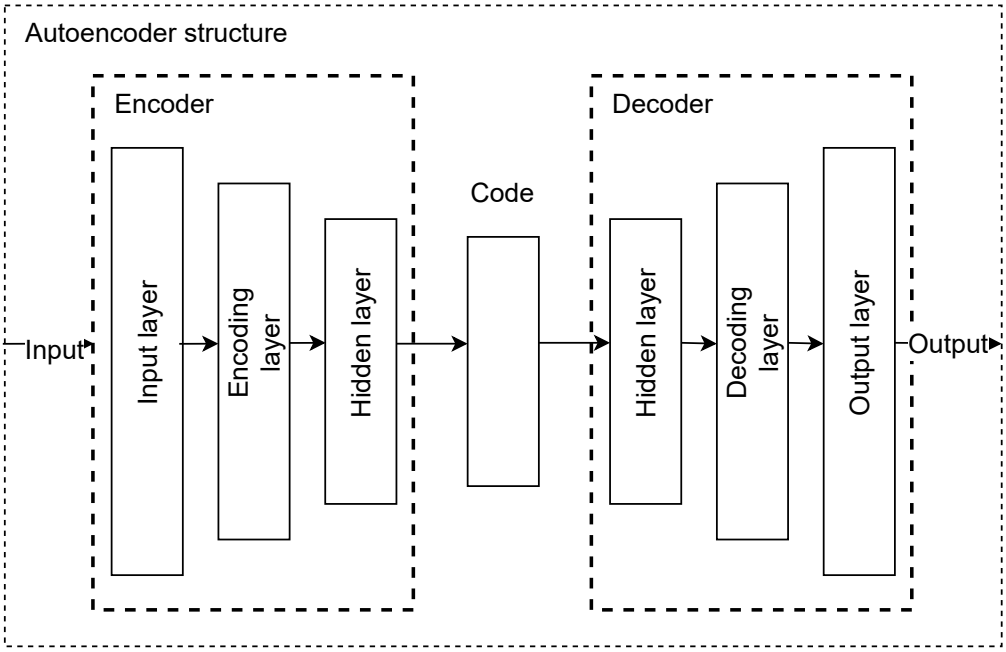

Figure 5: A network diagram of the Autoencoder used in our code

As illustrated in Figure 5, we built a neural network for autoencoder structure with three layers of the encoder, one latent space layer, and three layers in the decoder. The encoder layers were an input, an encoding, and a hidden layer. The

decoder layers were a hidden layer, a decoding layer, and an output layer. It is a symmetric design where the input and output layers have the same dimension d as the three data sets. The latent space layer is the bottleneck layer that needs to capture the data in lower dimensions.

To decide the number of nodes in these layers, we used a structure parameter α. α is the ratio from the previous layer that decides the number of nodes in the present layer. We calculated the nodes in the first encoding layer by dividing the input dimension d by 2, the hidden dimension was $d/4$, and the latent space dimension was $(d/4) - 2$. These parameters were carefully selected to maintain a generic structure for all the data sets. These nodes gave us optimal results.

Without activation, all the neural network layers give a linear function of the inputs. To incorporate the non-linear behaviour, we used activation functions elu for the encoder and relu for the decoder. These activation functions were selected after several experimental iterations with different activation functions. Exponential linear unit (ELU) does not cause vanishing gradient or exploding gradient problems (Hanin 2018). It is continuous and differentiable. It outputs the input; if the input is positive, and if the input is negative, the output is one less than the exponential of the input. This ensures that the output is closer to 1, even when the inputs are largely negative. A mathematical reference is as follows: (Singh 2021a) $y = \text{ELU}(x) = \exp(x) - 1$; if $x < 0$ $y = \text{ELU}(x) = x$; if $x \geq 0$. Rectified Linear Unit (RELU) compares the input to 0, and the output is the maximum between 0 and the input. Hence the output is always positive. This is a continuous function, but the derivative does not exist. A mathematical reference is as follows (Singh 2021b): $y = \text{RELU}(x) = \max(0, x)$.

4.2.2 Overview of ensemble learning using autoencoder

The above-explained autoencoder code ran 36 times to get 36 base models for our ensemble learning, as shown in Figure 6. We tested this model for 4 data sets - cardio, gas drift, letter and satellite. After careful iterations, an epoch count was 200, and a batch size was 64. Epoch indicates that the entire dataset is

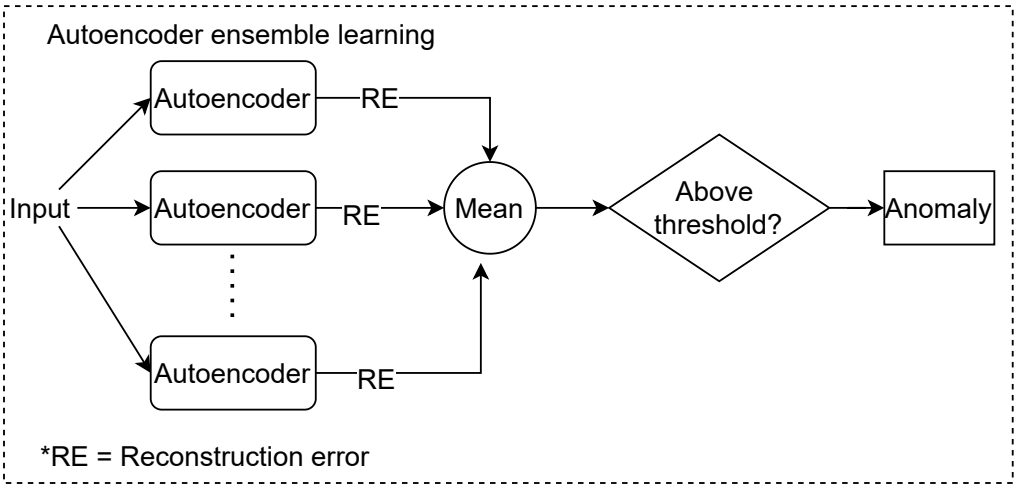

Figure 6: A flowchart of ensemble learning with autoencoders used in our code

passed forward and backwards through the data set 200 times. As the datasets are massive during training, in each epoch, we divided them into a smaller batch size of 64. Each base model autoencoder was trained on the training data, and the test data was used in prediction. After comparing the predicted output to the input, a mean squared error was calculated for each data point, as explained in Section 3.2. The mean squared error for a data point from each base autoencoder model was averaged. The data point was identified as an anomaly if this average mean squared error exceeded a threshold. Each possible threshold was evaluated to calculate the optimal threshold, and the one that gives a maximum ROC was selected. After we identified all the anomalies, the ROC graph was plotted to see the performance of our model. The results of this model can be seen in Section 5.1

4.3 Modified model for ASE learning

We modified the existing autoencoder ensemble learning model in the next step to get our novel ASE model. We initially started with the same 4 data sets mentioned above and expanded the experiment to 36 data sets.

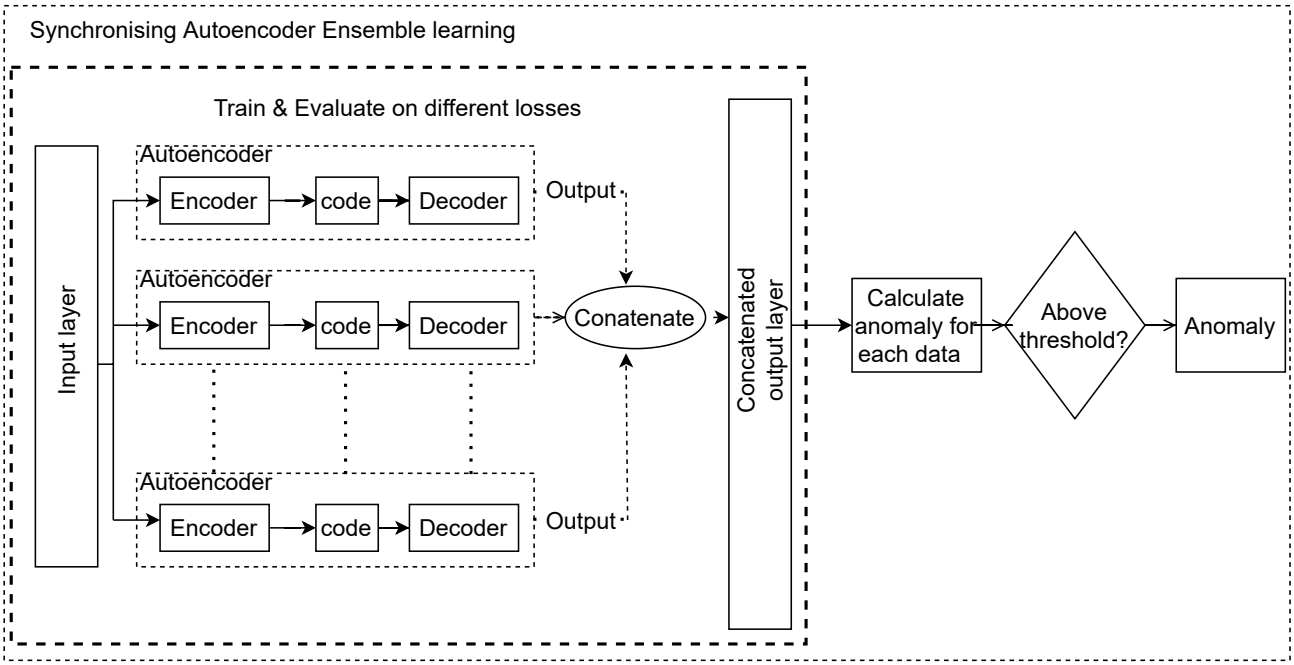

Figure 7: A flowchart of the novel synchronising autoencoder ensemble learning used in our code

4.3.1 Overview of the model

As shown in Figure 7, we have a single input layer with the same number of nodes as the number of features in the data set. Through this layer, multiple autoencoder base models were fed with the input. The number of autoencoder base models varied from 3 to 60, with a jump of 3. This means that we checked the performance of the ASE model with 3 autoencoder base models, then 6, 9, until 60. These autoencoders' structure and other parameters were the same as explained earlier in Section 4.2.1.

Once the data was encoded through the encoder, the compact representation was captured in the latent space. The decoder tried to output the reconstructed input, and every decoder output was concatenated. Hence there is a single output layer with concatenated outputs from every decoder. This concatenated output was N times the original input with an additional dimension. N is the number of synchronising autoencoder base models. This concatenated higher dimensional output was backpropagated with appropriate losses during the training to reduce the reconstruction errors. All the losses explained from Section 3.3 to 3.5 were used

sequentially to train our model. Once the model was trained, we did two kinds of evaluation, one with MSE loss and one with training loss itself. We evaluated with MSE loss, as that is how the general ensembles were evaluated. But as some of the losses are learnt differently than MSE, we also evaluated with the training loss to get the actual ASE model performance. Each of our losses is written so that they can compare the N times higher output with the test input and calculate the reconstruction error for the test data input. To avoid any fluctuations in ROC, we ran the whole experiment 5 times and averaged the ROCs we got. So, for each data set and loss, as we check each loss sequentially, we are building 100 ASE models; that is, 20 ASE models with varying base models run five times. With these ROC graphs, we checked the performance of the ASE model. These models' results are discussed in Section 5.2.

5 Results and discussion

This chapter summarises all the significant findings of both existing autoencoder ensemble learning and ASE learning. We individually discuss the results of the two kinds of model used, and finally compare the two methods based on ROC. The existing autoencoder ensemble learning ran for 4 data sets and the ASE model ran for 36 models.

5.1 Results of the existing autoencoder ensemble learning

We ran the aforementioned model explained in Section 4.2.2 on cardio, gas drift, satellite, and letter data to plot the ROCs. We plotted the ROC of each data set to check the performance of the model. We had 36 base autoencoder model for each data set and the training and evaluation was done with MSE loss (3.2). Figure 12 is an example for ROC graph plotted for Cardio data set. The parameters used in this model for each data set can be found in Table 2. Table 3 summarises the ROCs the general autoencoder ensemble model we got for each data set.

Table 2: Overview of the parameters of the general autoencoder ensemble model

Data set	**Encoding Dimension**	**Hidden Dimension**	**Latent Space Dimension**
Cardio	10	5	3
Gas drift	18	9	7
Letter	16	8	6
Satellite	64	32	30

Table 3: Overview of the results of the existing autoencoder ensemble model

Data set	**ROC value**
Cardio	0.94
Gas drift	0.92
Letter	0.68

Table 3: Overview of the results of the existing autoencoder ensemble model

Data set	ROC value
Satellite	0.77

5.2 Results of ASE model

The model explained in Section 4.3.1, ASE models were run 5 times, and the 5 ROCs we get for the 20 different ASE models (with 3 to 60 autoencoder base models) were averaged. We will have two variations of this ROC graph for each data set. One is when we evaluate with MSE loss, and the other is when we evaluate with the training losses. We evaluated with MSE loss as that is how we evaluated the general ensembles. But as some of the losses are learnt differently than MSE, we also evaluated with the training loss to get the actual ASE model performance. An example of ROC graphs plotted for one of the datasets, letter, can be seen in Section 5.2.1. Next, we also checked how ROCs varied each time. In short, the standard deviations among the ROCs in each ASE model for the 5 runs were calculated and plotted. This standard deviation is elaborately explained for one of the example data sets, letter, in Section 5.2.2. The correlation was one of the main factors we considered in building the ASE model. We wanted base models that are not correlated. Thus section 5.2.3 explains how we checked the correlation for our example letter data set and how it extends to other data sets. Finally, in Section 5.2.4, we get an overview of how each loss performed in our 36 data sets and compare them with the general autoencoder ensemble learning performance.

5.2.1 Analysis of the ROC graphs

The ROC graph displayed in Figure 8 represents the ROCs we calculated for one of the data sets - Letter. The model was evaluated with MSE loss in this figure. The letter is a data set with 1400 training data of 32 features. The test data had 100 anomalies. The x-axis represents the number of base autoencoder models.

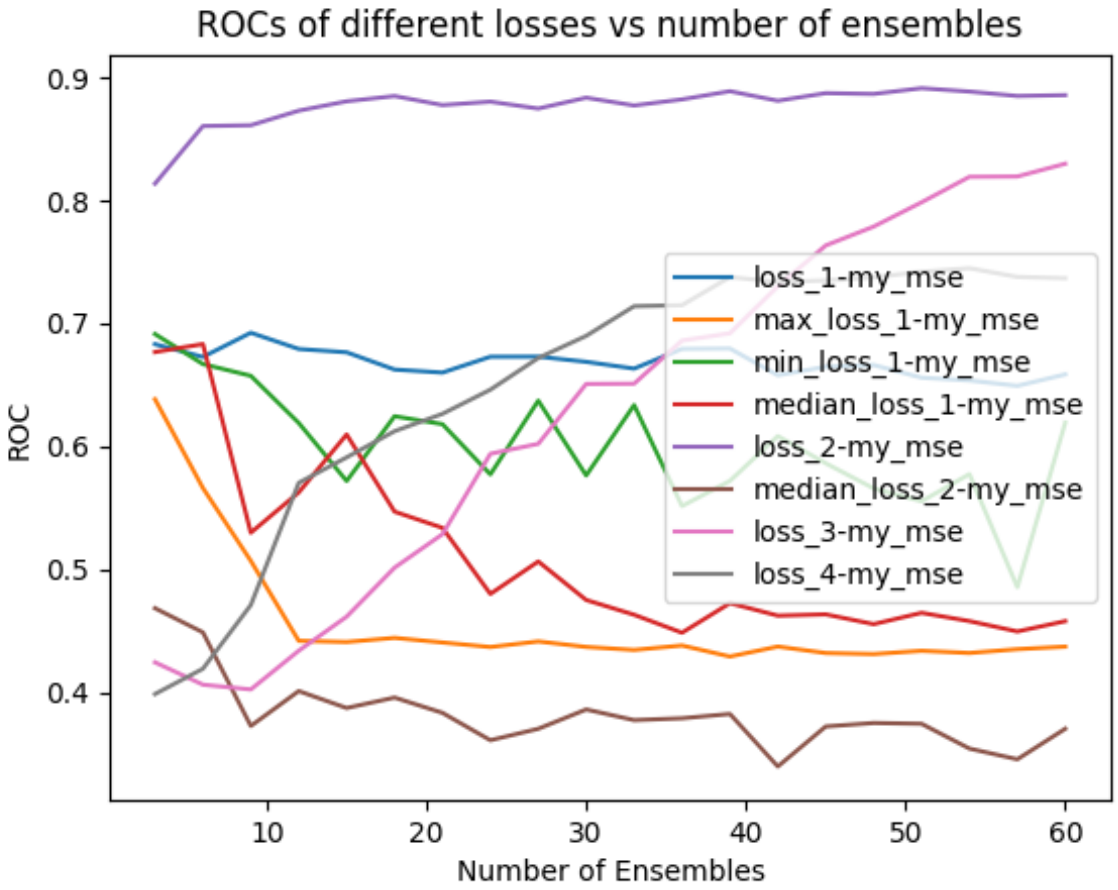

Figure 8: ROC plot of letter data (with MSE loss evaluation)

The y-axis indicates the corresponding ROC value. The ASE model was trained on various losses and evaluated on MSE loss. Thus we have a different ROC line for each loss. As explained earlier, we built 20 models with 3 base autoencoder models to 60 base autoencoder models. Every ROC line of each loss is also an average of 5 overall runs. For example, if we consider the 30 ensemble in the x-axis of Figure 8, we see 0.69 in the blue line. To be precise, when we ran our entire ASE model 5 times and averaged every ROC value we got for loss_1 with 30 base autoencoder models, the average ROC for loss was 0.69. The graph's legend shows that a different colour indicates every loss.

For the letter data set, we can observe that loss_2 is the best-performing loss. In contrast, every other loss shows an average performance below 0.7 ROC. We had explained that Loss_1 is the same as general autoencoder ensemble learning. We

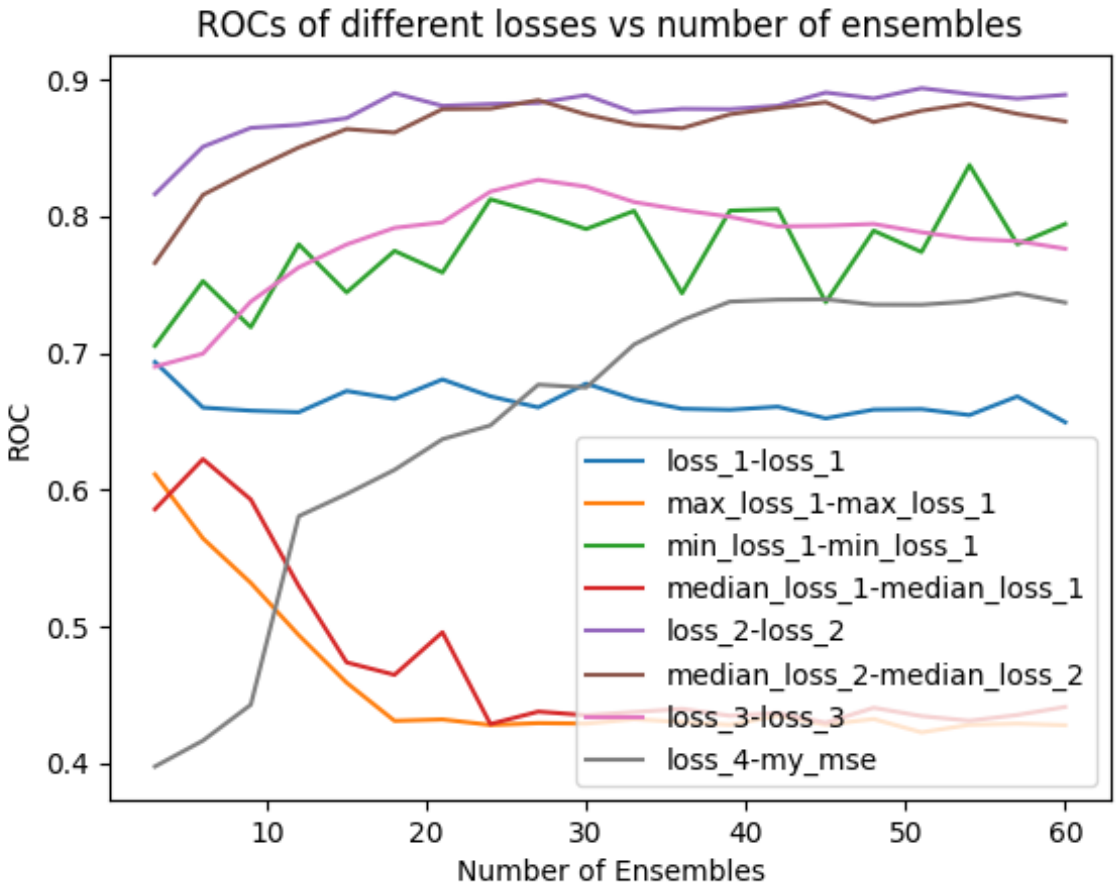

Figure 9: ROC plot of letter data (with training loss evaluation)

can see that from the outputs. In Table 3, we got an ROC of 0.68 for general autoencoder ensemble with 36 base models. Even in Figure 8, we can see that loss_1 has an ROC value of 0.68 with around 36 ensembles. outputs a stable ROC value of around 0.7. After approximately 27 autoencoder base models, loss_3 and loss_4 surpass loss_1, indicating that the interactions between the base autoencoders are helping. The rest of the losses are giving a poor performance compared to loss_1. We can also observe from Figure 8 that the variations of ROCs are higher when the number of autoencoder base models is low. As the base models increase, the ROC stabilises. This behaviour is expected because a few lousy base models do not affect the performance with the increase in the base models.

Some losses perform poorly when we evaluate with MSE loss. Hence we use the same loss used in training for evaluation. Please note that we cannot use loss_4 for evaluation as the data point dimensions are not available per the calculations of loss_4. However, we have still added loss_4 with MSE evaluation in this graph to see how it performs with respect to other losses. The ROC graph displayed in Figure 9 represents the ROCs we calculated for the letter data set when evaluating the model with the training loss. The graph is similar to the MSE loss evaluation, but the median loss_2 drastically improved. Earlier, the ROC for this loss was below 0.5 and performed worse than a random predictor. But now the ROC reached almost 0.9, one of the best-performing losses. We can also see a drastic ROC increase in loss_3 and min_loss_1. Loss_1 is unaffected as it is similar to the MSE loss of general autoencoder ensemble learning.

5.2.2 Analysis of the standard deviations among ROC graph plots

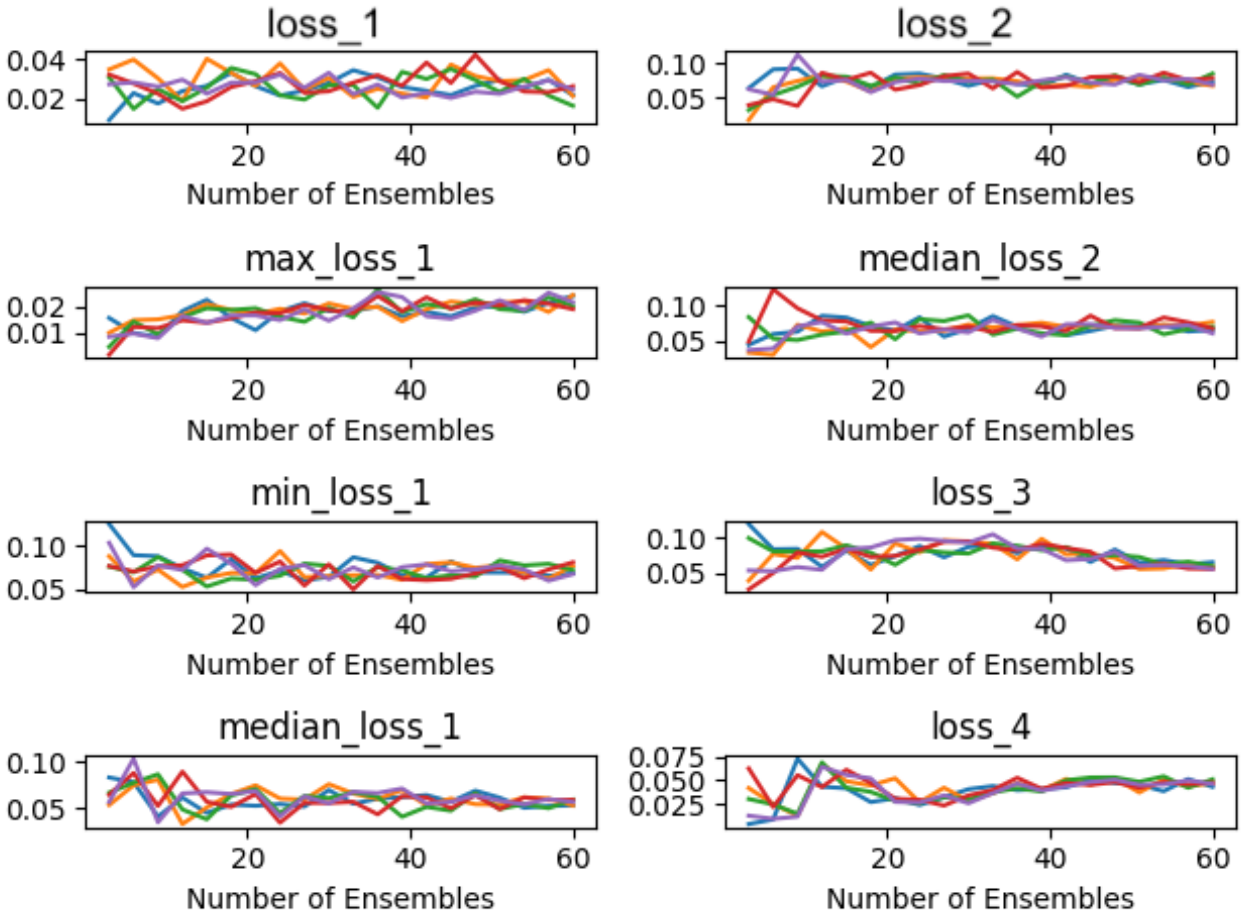

Figure 10: ROC standard deviations of letter data (with MSE evaluation)

Figure 10 shows the plot of standard deviations of ROCs generated by each synchronising base autoencoder model. The x-axis represents the number of base models, and the y-axis is the average standard deviation of ROCs over 5 runs in each ASE model with varying numbers of base models.

$$\mathrm{SD} = \sqrt{\frac{\Sigma\,|x-\mu|^2}{N}} \tag{12}$$

As shown in the formula, SD is the standard deviation, x is the ROC value from one base autoencoder model, μ is the average ROC, and N is the total number of base autoencoder models. The graph is plotted for each loss. There are five lines for each loss, indicating the sequential runs of the ASE model. To calculate this standard deviation, we only consider MSE evaluation as we want the individual result of each base auto-encoder model. But in other losses, the evaluations are done after factoring in the contributions of another base model. To calculate this standard deviation, we first train the entire ASE model and then calculate the reconstruction error of each base autoencoder model. After that, the ROC we get for each base autoencoder model is recorded. Then the standard deviations between the ROCSs of these base models are checked. As shown in Figure 10, the standard deviations are always less than 0.1, stabilising with the increase in base models. Thus we can tell that we get more reliable predictions as the number of base autoencoder models increases. We can also see that max_loss_1 has lower standard deviation than other variations of loss_1 like min_loss_1 or median_loss_1. Loss_2 and median_loss_2 had similar standard deviation between the ROCs. Loss_4 has the least standard deviation. This trend is observed in the rest of the data sets as well.

5.2.3 Correlation between ROCs for each loss

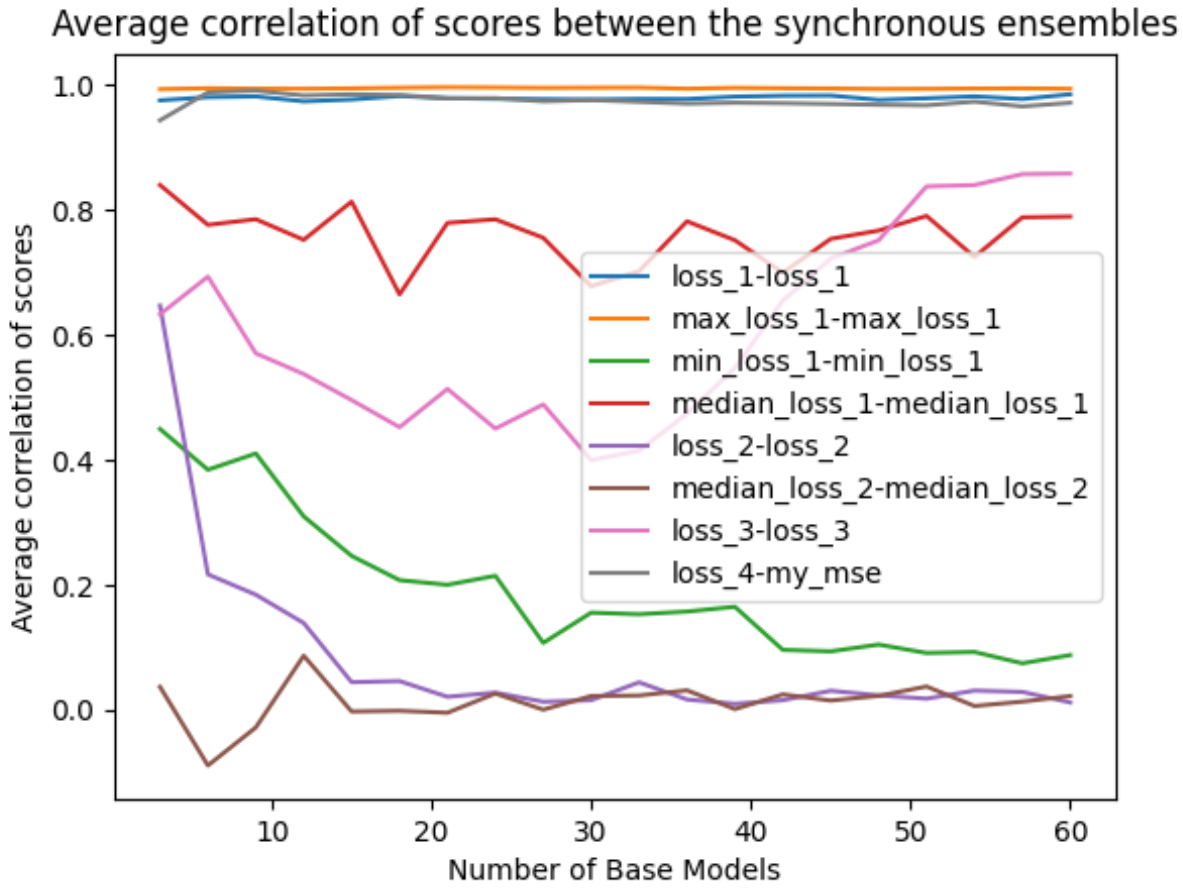

Figure 11: Correlations in the result of base autoencoder models in letter data

Figure 11 shows the correlations plot in the synchronising base autoencoder model results. The x-axis represents the number of base models, and the y-axis represents the correlation between the scores of the base autoencoder model for each of the losses in the letter data set. Similar to the standard deviation in the previous section, we only consider MSE evaluation to calculate correlation as we want the individual result of each base auto-encoder model. We first train the entire ASE model and then calculate the reconstruction error of each base autoencoder model. The whole model was run 5 times, and the reconstruction error was averaged.

Then, the correlation was calculated across the score generated by the individual base models. The correlation of the base model with itself was excluded from this calculation as it is always 1. This step was repeated for all 20 models with varying base models from 3 to 60. These average correlation values for each loss function were plotted and compared as seen in Figure 11. A similar trend was observed for all the data sets. Loss_1 had a correlation of 1, and our ASE model losses had a lower correlation. Losses like loss_2, loss_3, and median_loss_1 that performs well had low correlation. Loss_4 had a high correlation in a few data sets and a low correlation in a few more.

5.2.4 Comparison and comments on each loss of ASE model

In Section 5.2.1, we compared the ROC lines of different loss functions for a single letter data set. We have 72 ROC graphs, 36 with MSE loss evaluation and 36 with training loss evaluation for all the 36 data sets used in this experiment. This section analyses the results in the ROC graphs of all the data sets and compares the losses' performances. The benchmark to compare the losses' performances is loss_1, which is the same as the performance of a general autoencoder ensemble model.

To compare the ROC values of all the losses with loss_1, each ROC curve is subtracted from loss_1, and the difference of each loss curve is averaged. The difference is mostly taken after 20 base models. This is because the initial results with few models had a lot of noise. This value is called the mean difference denoted by l_d. For example, to calculate l_d between loss_4 and loss_1 in Figure 8, the purple loss_4 curve is subtracted from the blue loss_1 curve. Then, the difference at all the points is averaged. In the next step, a p-value for each loss is obtained by performing a Wilcoxon's test of every loss curve against loss_1. This test indicates if the loss curves are significantly different than loss_1.

A non-parametric test called the Wilcoxon test determines whether there is a significant difference between the means of two dependent groups (DATAtab 2022a). The null hypothesis of the Wilcoxon test is that the two groups are significantly

similar, and the alternate hypothesis is that they are not similar. The null hypothesis is either rejected or not rejected based on the p-value. The null hypothesis is not rejected if the p-value is less than the set significance level, which is typically 5% (DATAtab 2022b).

Based on the value of l_d and p-value, losses are categorized into 3 groups for every data set. If the l_d value is greater than 0, and the p-value is greater than 0.05 for a particular loss, the selected loss is significantly better than loss_1. This implies that our ASE model performs better than the general autoencoder ensemble model. If the l_d value is lesser than 0, and the p-value is greater than 0.05 for a particular loss, it indicates that the selected loss is significantly worse than loss_1. This implies that the general autoencoder ensemble model is better than ASE, and our novel algorithm may not be effective. If the p-value is less than or equal to 0.05, the selected loss is the same as loss_1, indicating that ASE is neither helping nor making the predictions worse.

Table 4 and 7 indicate the number of data sets in each of the three categories for every loss. Table 4 is when we evaluate the model performance with MSE loss, and Table 7 is when we evaluate the model performance with the training loss. We used 36 data sets to categorize the losses into the best, same and the worst category compared to loss_1. However, the musk data set had a ROC of 1 for every loss. It was the most straightforward data to learn, with evident anomalies. Thus Musk data set is not used in the table as we do not have anything to test in this data. Out of 35 data sets, we can see that loss_1 and loss_2 are performing better in the highest number of data sets when we evaluate with MSE loss. min_loss_1 is the best-performing loss in the highest number of data sets when we evaluate the model with the training loss. These results also confirm our assumption that few losses would perform better if we evaluate with the same loss. min_loss_1 is among those losses as min_loss_1 would always learn the best model with the minimum loss, and evaluating it with an MSE loss where we average all the models would be different. loss_4 is performing well in both kinds of evaluation. It is a loss that consistently performs well. Table 4 and 5 show the sum of data sets in best loss and the significantly non-different losses categories.

Out of 35 data sets, we can see that more than 60% of data sets are performing better or the same as loss_1 in Table 4 for loss_3 and loss_4. Hence using ASE models with these losses is improving the model performance. Out of 35 data sets, we can see that more than 60% of data sets are performing better or the same as loss_1 in Table 5 for loss_3, loss_4, and also min_loss_1. Hence using ASE models with these losses improves the model performance when we evaluate with training loss. Thus we can tell that our novel ASE model can find meaningful anomalies by factoring in the interactions of the model in losses like loss_4, loss_3 and min_loss_1. But losses like max_loss_1, median loss_1 and median loss_2 work only for specific data sets. So we cannot ensure better results for most data sets with these losses.

Table 4: Overview of each loss performance (with MSE loss evaluation)

Loss name	**Data sets with non-worse performance**
loss_2	22
loss_3	21
loss_4	26
max_loss_1	18
median_loss_1	18
median_loss_2	17
min_loss_1	12

Loss name	**Data sets with non-worse performance**
loss_2	18
loss_3	21
loss_4	24
max_loss_1	14
median_loss_1	16

Loss name	**Data sets with non-worse performance**
median_loss_2	17
min_loss_1	23

6 Conclusion and future work

Anomaly detection refers to finding a subset of a particular data set that behaves differently compared to the rest of the data points. Unsupervised anomaly detection is beneficial since most real-world data sets are unlabelled. Autoencoders are a powerful unsupervised anomaly detection network with an encoder, latent space, and decoder. The encoder receives the input and compresses it in the latent space. The decoder attempts to recreate the input out of the compressed latent space. When this network encounters an anomaly, the reconstruction error is higher for anomalous data. Hence, we identify the anomalies. But instead of building a single autoencoder model and hoping for a good performance, we can build several models and combine them. This concept is ensemble learning, where we build several weak base models, combine their results, and get a powerful model. But the problem is when we use several autoencoders in ensemble learning, they all tend to give similar results because they all reconstruct the same input space. Therefore, combining similar autoencoders will not provide any significant results. The variation will be low, and the correlation will be high. In ensemble learning, we need diverse results to capture different input spaces and get a good result. This thesis aims to improve autoencoder ensemble learning by considering the interactions between the base autoencoder models. Thus we build a novel ASE model and test it on 36 data sets.

A new approach called ASE (Autoencoder Synchronising Ensemble) addresses the problem of base auto encoders having high correlation and low variance. The problem could be solved by considering how the underlying models interact. Additionally, the issue of combining non-diverse results is solved as ASE concatenates the output of every base model. Losses that result from interactions between all the basic models are decreased through backpropagation. The ASE algorithm's losses, which considers how one autoencoder model affects another, are its distinctive characteristic. In ASE, we attempt to reduce different losses affecting the concatenated base autoencoder output. The losses that we consider are loss_1, min_loss_1, max_loss_1, median_loss_1, loss_2, min_loss_2, max_loss_2,

median_loss_2, loss_3 and loss_4. The loss_1 is similar to the MSE loss of general autoencoder ensemble learning. The only difference is that we consider the MSE loss between higher dimensional outputs. The min_loss_1, max_loss_1, and median_loss_1 are modifications to this higher dimensional MSE loss where we consider only the best, worst and median performing models, respectively, instead of averaging across all the models. The loss_2 is where we first average the predictions of all the synchronizing ensembles and then find an MSE loss between this average prediction and real output. Thus we factor in the synchronizing base model outputs before optimizing the network. The min_loss_2, max_loss_2, and median_loss_2 are the modifications to this loss_2 where we consider the models with the highest, lowest and median values of prediction. The concept of covariance inspires loss_3. Here we consider the effect of one model prediction on the other for a particular feature. Loss_3 inspires the loss_4. Instead of considering the effect of one model over the other for each feature, we average mean square error across all the features for one base model and see how it varies with the average mean square error across all the features for another base model.

Once we code the above-explained ASE model with various losses that consider the interactions between the base models, we verify this for 36 data sets. We build 20 different ASE models with 3 to 60 base models. That is, the ASE model with 3 base models, 6 base models, 9 base models, and so on until 60 base models. All these 20 models are run 5 times, and the 5 different results are averaged to avoid model noises. In short, 20 base models are run 5 times for each loss, and the average ROCs are plotted for each data set. We have two versions of this plot. The first is when we evaluate the model with MSE loss, and the next is when we evaluate the model with the training loss. We evaluate with MSE loss, as that is how it is done in the general autoencoder ensemble. The comparison becomes easier. But as some of our losses produce models different from MSE loss, we also evaluate with the training loss. We cannot evaluate loss_4 with loss_4, as we average across the data points in this loss. Thus loss_4 is always evaluated with MSE loss. In the two ROC graphs, we get with two evaluations, we check the losses with good ROC values. We also check the standard deviation of these ROC

values. We see that the model stabilizes, and the standard deviation decreases as the number of base models increase. Our following requirement was a low correlation. We check the correlation between the base models next. Loss_1 has a high correlation of 1 as we would have expected, since it is the same as the general autoencoder ensemble. The losses of the ASE model show a lower correlation in all the data sets. Even though a few losses like loss_4 show a higher correlation in some of the data sets, we can generally observe a lower correlation for ASE model losses. Lastly, we categorize all the losses in 36 data sets into 3 categories to check which losses are effective. To categorize the data set, we compare the ROC values of every loss with that of loss_1. Hence we can check if our losses are performing better than loss_1, that is, the general autoencoder ensemble. The 3 categories are the best loss category where the losses are significantly performing better than loss_1, the worst loss category where the losses are significantly performing worse than loss_1, and the same category where a particular loss and loss_1 has the same performance.

When we evaluate the model with MSE loss, we see that 60% to 70% of the data sets, loss_2, loss_3 and loss_4, are performing better or similar to loss_1. When we evaluated the model with training loss, we also saw that ASE performed better than loss_1 for min_loss_1 and loss_3. Losses like max_loss_1, median_loss_1, and median_loss_2 also performed well for specific data sets. The losses min_loss_2 and max_loss_2 are dropped from the plotting and calculations as they were the worst-performing losses. Hence we can tell that loss_3 and min_loss_1 can be good choices for ASE models, especially when evaluating with training loss. The loss_4 has good performance but may have a high correlation for a few data sets. The loss_2 can be a good choice when we evaluate with MSE loss, and it also has a low correlation. But after understanding the data better, we can run the experiments and select a suitable loss function that works better than the general ensemble model and gives high base model variance and low correlation.

Further studies can be first in hyperparameter tuning, which is mentioned in Section 2.2.2. Iterating the experiments several times or performing cross-validation

to decide different node sizes, layer, seed, epoch or batch sizes can be highly effective. The code of our thesis is written for the Cardio data set first. All the hyperparameters are tuned for this data set. As a result, every loss of the Cardio data set performs better than loss_1. This result may indicate that hyperparameter tuning can effectively get good results. But we did not do this for all other 35 data sets due to a time crunch. Secondly, we can investigate why a good performing loss like loss_4 shows a high correlation on a few data sets. Next, this thesis considered only ROC as a measure to evaluate performance. Further, parameters like accuracy, precision, and F-scores can reveal additional details about our ASE model's effectiveness. Also, to consider more interactions between the models, there is potential to define many new losses. We can also consider different ensemble learning concepts like boosting. Even though boosting is impossible to be considered with anomaly detection, as we are dealing with unsupervised correlation here, we could define new loss functions that can employ boosting. Lastly, loss_4 or loss_3 perform well in the ASE model but are slightly complex to understand. We can also research more straightforward loss functions that give similar results by considering the model interactions.

References

[1] Martn Abadi et al. *Tensorflow: A system for large-scale machine learning*. New York, USA: arXiv.org, 2016.

[2] Hmrishav Bandyopadhyay. *Autoencoders in Deep Learning: Tutorial & Use Cases*. 2022. URL: https://www.v7labs.com/blog/autoencoders-guide#h (visited on 10/02/2022).

[3] Jason Brownlee. *Autoencoders in Deep Learning: Tutorial & Use Cases*. 2021. URL: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (visited on 10/02/2022).

[4] Jinghui Chen et al. "Outlier detection with autoencoder ensembles". In: (2017).

[5] CorporateFinance. *Ensemble Methods, Combining multiple models to improve the desired results*. 2021. URL: https://corporatefinanceinstitute.com/resources/knowledge/other/ensemble-methods/ (visited on 10/02/2022).

[6] Casper O. da Costa-Luis. "tqdm‘: A Fast, Extensible Progress Meter for Python and CLI". In: (2019).

[7] DATAtab. *What is the p-value?* 2022. URL: https://datatab.net/tutorial/wilcoxon-test (visited on 10/02/2022).

[8] DATAtab. *Wilcoxon-Test*. 2022. URL: https://datatab.net/tutorial/p-value (visited on 10/02/2022).

[9] TU Delft. *One-class classifier results*. 2022. URL: http://homepage.tudelft.nl/n9d04/occ/ (visited on 10/12/2022).

[10] Arden Dertat. *Applied Deep Learning - Part 3: Autoencoders*. 2017. URL: https://medium.com/towards-data-science/applied-deep-learning-part-3-autoencoders-1c083af4d798 (visited on 10/02/2022).

[11] Ted Dunning and Ellen Friedman. *Practical machine learning : a new look at anomaly detection*. Sebastopol, CA: O
Reilly Media, 2014.

[12] Jim Frost. *Introduction to Statistics: An Intuitive Guide*. Online: Statistics By Jim, 2019.

[13] Yulia Gavrilova. *What Is Anomaly Detection in Machine Learning?* 2021. URL: https://serokell.io/blog/anomaly-detection-in-machine-learning (visited on 10/02/2022).

[14] Boris Hanin. "Which Neural Net Architectures Give Rise To Exploding and Vanishing Gradients?" In: (2018).

[15] J.A. Hanley and Barbar Mcneil. "The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve". In: (1982).

[16] Charles R. Harris et al. "Array programming with NumPy". In: (2020).

[17] Chengqiangi Huang. "Featured anomaly detection methods and applications". In: *University of Exeter* (2018).

[18] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: (2007).

[19] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. "Performance anomaly detection and bottleneck identification". In: (2015).

[20] Simon Kluettermann. *yano 0.92*. 2022. URL: https://pypi.org/project/yano/.

[21] Simon Klüttermann and Emmanuel Müller. "DEAN: Deep Ensemble Anomaly Detection". In: (2022).

[22] Wes McKinney et al. "Data structures for statistical computing in python". In: (2010).

[23] James Moody. *The Geometric Meaning of Covariance*. 2019. URL: https://towardsdatascience.com/the-geometric-meaning-of-covariance-f8e6df967111 (visited on 10/22/2022).

[24] Dalmas Otieno. *Introduction to ensemble methods*. 2018. URL: https://medium.com/@dalmas.otieno/introduction-to-ensemble-methods-aca988f25fcb (visited on 10/02/2022).

[25] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: (2011).

[26] Shebuti Rayana. *ODDS Library*. 2016. URL: http://odds.cs.stonybrook.edu.

[27] Robert E Schapire. *Explaining adaboost*. Gewerbestrasse 11, 6330 Cham, Switzerland: Springer, 2013.

[28] Sinch_blog. *Dynamic threshold estimation for anomaly detection*. 2021. URL: https://www.sinch.com/blog/dynamic-threshold-estimation-for-anomaly-detection/ (visited on 10/02/2022).

[29] Saurabh Singh. *ELU as an Activation Function in Neural Networks*. 2021. URL: https://deeplearninguniversity.com/elu-as-an-activation-function-in-neural-networks/ (visited on 10/02/2022).

[30] Saurabh Singh. *ELU as an Activation Function in Neural Networks*. 2021. URL: https://deeplearninguniversity.com/relu-as-an-activation-function-in-neural-networks/ (visited on 10/02/2022).

[31] Statlog. *Statlog*. 2022. URL: https://www.statlog.co.uk/ (visited on 10/02/2022).

[32] ACM computing surveys. *OpenML: Networked Science in Machine Learning*. 2013. URL: https://new.openml.org/ (visited on 10/12/2022).

[33] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[34] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: (2020).

[35] Michael L. Waskom. "seaborn: statistical data visualization". In: (2021).

[36] Wikipedia. *MNIST database*. 2022. URL: https://en.wikipedia.org/wiki/MNIST_database (visited on 10/02/2022).

[37] Wikipedia. *Receiver operating characteristic*. 2022. URL: https://en.wikipedia.org/wiki/Receiver_operating_characteristic (visited on 10/02/2022).

[38] Arthur Zimek, Ricardo JGB Campello, and Jörg Sander. "Ensembles for unsupervised outlier detection challenges and research questions a position paper". In: (2014).

Appendix

A Additional figures

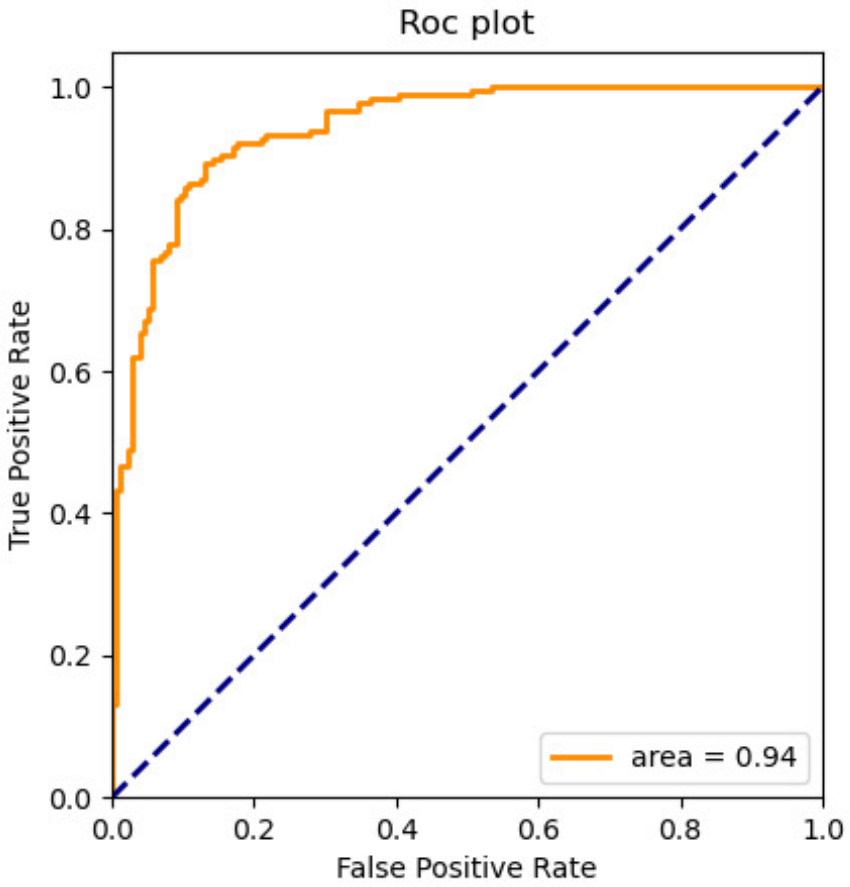

Figure 12: ROC plot of cardio data

B Additional tables

Table 6: Performance of the losses in all the data sets compared to loss_1 (with MSE evaluation)

Loss name	**Data sets where the loss is better**	**Data sets where the loss is the same**	**Data sets where the loss is worst**
loss_2	14	8	13
loss_3	10	11	14
loss_4	14	12	9
max_loss_1	9	9	17
median_loss_1	8	10	17
median_loss_2	9	8	18
min_loss_1	6	6	23
None	1	1	1

Table 7: Performance of the losses in all the data sets compared to loss_1 (with training evaluation)

Loss name	**Data sets where the loss is better**	**Data sets where the loss is the same**	**Data sets where the loss is worst**
loss_2	13	5	15
loss_3	13	8	12
loss_4	13	11	9
max_loss_1	5	9	19
median_loss_1	7	9	17
median_loss_2	11	6	16
min_loss_1	16	7	10
None	1	1	1

C Details of each data set

- **Arrythmia:** Arrythmia is a condition with abnormal heartbeat, where the heart beats faster or slower than usual. The UCI's (The University of California Irvine Machine Learning Repository) Arrhythmia database is a multivariate data set with the aim to diagnose Arrythmia and categorise into one of the 16 groups indicated by the cardiac experts (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Cardio:** The UCI's (The University of California Irvine Machine Learning Repository) Cardio dataset includes Fetal Heart Rate (FHR) and Uterine Contraction (UC) measurements on cardiotocograms* that are classified into normal, suspected and pathological classes by experienced obstetricians and gynaecologists.To detect outliers, the suspected group is removed and the normal group constitutes the outlier, and the pathological (outlier) group is reduced to 176 points.
  *Cardiotocography (CTG) is the measure of a baby's heart rate (Kluettermann 2022, Rayana 2016).
- **Concardia3_32:** Concordia dataset is a dataset where the digits are stored in 32»32 black-and-white images. Each digit is a class in itself. When we train the model with a particular digit, the rest of the Concordia digit boxes are outliers. In our experiment, we used the Concordia data set with the digit 3 (Kluettermann 2022, Rayana 2016).
- **Delftpump_5X3_noisy:** This data set is from the Delft university. From the submerged pump at Delft, 5 vibrations are measured at various normal and abnormal conditions. The time signals are the used get a 64 dimension of envelope spectrum and the 5 measurements from normal situations are used as independent objects in classification against abnormal conditions (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Delftpump_AR:** Ts data set is also from the Delft university where the 5 vibrations are measured at various normal and abnormal conditions from the submerged pump at Delft. Unlike the previous data set, an autoregressive

model is fitted on this data set to give 32 features and 5 measurements are combined to give an 160D feature space (Kluettermann 2022, Delft 2022, Rayana 2016).

- **Elevators:** This data set was gathered while operating an F16 aircraft. The target variable is connected to a movement on the aeroplane's elevators. This objective has been binarized. By finding the mean and identifying all cases with a lower target value as positive ('P') and all other instances as negative ('N'), it transforms the numeric target feature into a two-class nominal target feature.
- **Gas drift:** This data set contains data from 16 chemical sensors that give 13910 measurements in the simulation to compensate for drift in the problem of discriminating 6 gases at different concentrations. Over the time, a good performance that indicates less degradation is the goal. (Kluettermann 2022, surveys 2013, Rayana 2016).
- **HeartC:** The 13 selected attributes are the heart disease indicators from Cleveland database. If the patient suffers from a heart disease, it is indicated in the target field. Patient identification is anonymous and signs of heart disease are integers between 0 (no heart disease) to 4. The aim is to separate the presence (values 1, 2, 3, 4) of the heart disease the from absence (value 0) (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Hepatitis:** Hepatitis is an inflammation of the liver and hepatitis A, B, C, D, and E are the five major hepatitis viruses. These diseases are dangerous because of their potential for outbreaks and the risk of death they cause. Most people with hepatitis are often asymptomatic, and doctors often use laboratory tests, such as liver function tests, prothrombin time tests, biopsies, or a physical examination of the person to diagnose the disease. The goal of this project is to create a predictive model that can help clinicians determine the prognosis and survival of patients with hepatitis based on given health characteristics (Kluettermann 2022, Delft 2022, Rayana 2016).

- **Hill valley:** The Hill Valley dataset has 100 points in a 2D plot, and plotting them in order from 1 to 100 produces either a hill, that is a bump, or a valley, that is a dip (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Housing low:** The Boston housing database is from the Statlib library and it is used to predict home prices in Boston suburbs with an average price of less than $\$35,000$ (Kluettermann 2022, Delft 2022).
- **Imports:** The import data set is pulled from UCI's vehicle database and contains two classes, each defined as vehicles with an assigned insurance risk class (first feature) greater than or less than (and equal to) zero respectively. Low risk is the target class (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Ionosphere:** The UCI machine learning repository has Ionosphere data set that is a binary classification data with 34 dimensions. After the removal of one of the non-contributing attribute, the current data set has 33 dimensions. The ionosphere data is from a radar set by the Goose bay laboratory. There were two kinds of radar data from this setup, good radar data with evidence of some structure in the ionosphere and bad radar data with no indication of any ionosphere structure. The outliers are the bad radar data with no structure (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Letter:** This data set has 26 different capital letters of alphabets of the English language in rectangular black and white pixel displays. The training goal is to identify the English alphabet in the data set. The data sets are generated by randomly distorting the alphabets from 20 fonts. The distortions of 20 fonts create 20,000 unique samples. Each sample is defined by 16 numerical attributes like edge counts and statistical moments. These attributes are scaled to have values from 0 to 15. This data set with 16 attributes is used in training to recognise the letters (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Magic telescope:** The Monte Carlo program generates this data set, and the imaging technique is used in a Cherenkov gamma telescope to simulate the high-energy gamma particle registration. By using the radiation

given off by the charged particles formed inside the electromagnetic showers that the gamma rays trigger and grow in the atmosphere, the Cherenkov gamma telescope can see high-energy gamma rays. This Cherenkov radiation, which has wavelengths ranging from visible to UV, escapes through the atmosphere and is detected, allowing the characteristics of the shower to be reconstructed. Pulses that the approaching Cherenkov photons left on the photomultiplier tubes placed in a plane and the camera make up the information that is now accessible. To statistically distinguish between hadronic showers created by cosmic rays in the high atmosphere (background) and those caused by primary gamma rays (signal), a few hundred to around 10000 Cherenkov photons are collected in patterns (Kluettermann 2022, surveys 2013, Rayana 2016).

- **Mnist:** The National Institute of Standards and Technology had a database with handwritten digits taken from employees of the ACB (American Census Bureau) and high school students of America. The data was not diverse enough to be used in machine learning problems, and it had some grayscale level problems due to the normalisation of the black and white images. By remixing this data set, we have the current MNIST database (Modified National Institute of Standards and Technology database) that are used in machine learning problems (Kluettermann 2022, Rayana 2016, Wikipedia 2022a)
- **Musk:** The goal of this dataset is to discriminate the musk molecules from the non-musk molecules. The dataset has 92 molecules and 166 features like shape, conformation, etc. that describe the molecule. Experts recognised 47 out of the 96 molecules to be musks. As the molecular bonds rotate, all of these molecules have different shapes. Hence a lot of low-energy conformations of each molecule were generated to create this data set. A feature vector was then extracted, to identify each conformation. IWhen we train the model, if any of the molecule conformations were recognised as musk, the molecule is of a musk. If none of the conformations of a molecule

is a musk, the molecule is identified as a non-musk (Kluettermann 2022, surveys 2013, Rayana 2016).

- **Opt digits:** As explained earlier, the National Institute of Standards and Technology had a database with handwritten digits taken from employees of the ACB (American Census Bureau) and high school students of America. The normalised $32X32$ bitmaps of these data set were extracted by 43 people and divided into $4x4$ blocks and the pixels on each block were counted. This data was then used to train the model to recognise the digits (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Ozone level 8hr:** The eight-hour ozone peak set from 1998 to 2004 in the Houston, Galveston and Brazoria area constitutes the data set. It has several attributes like temperature, relative humidity, geopotential height, sea level pressure, and many more. It is a binary classification problem to indicate non-ozone day as a normal data set and polluted day as an anomaly (Kluettermann 2022, surveys 2013, Rayana 2016).
- **PC3:** The goal of this data set is software defect detection in the flight software of NASA. This flight software is for satellites orbiting the earth, and it characterises the code features associated with the quality of the software (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Pen digits:** This data set was compiled using 250 samples from 44 writers and is available in the UCI machine learning repository. Instructions are given to these authors to fill boxes with a resolution of 500 by 500 tablet pixels using 250 digits in a random sequence. Each screen has five boxes with the required digits shown above them. The writers need to write solely inside these boxes. The examples produced by the other 14 authors are utilized for writer-independent testing. In contrast, the samples written by the other 30 writers are used for training, cross-validation, and writer-dependent testing (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Qsar biodeg:** The QSAR biodegradation data set was generated by Milano Chemometrics and the QSAR research group. The data were used to

develop a Quantitative Structure-Activity Relationship (QSAR) model to study the relationship between chemical structure and biodegradation of molecules. Experimental biodegradation values of 1055 chemicals were collected from the National Institute of Technology and Evaluation of Japan (NITE) web page. A classification model was developed to distinguish between ready (356) and not ready (699) biodegradable molecules. Using a data set containing values for 41 properties (molecular descriptors), 1055 chemicals were classified into two classes (readily and not readily biodegradable) (Kluettermann 2022, surveys 2013, Rayana 2016).

- **Satellite:** The initial satellite data for this database were obtained from data obtained by NASA at the Australian Remote Sensing Center. The data set we are using is a sample database created by taking small chunks (82 rows and 100 columns) from the original data. The database consists of multi spectral pixel values of 3x3 satellite image neighbourhoods and classifications associated with the central pixel of each neighbourhood. The goal is to predict classification given multiple spectral values. In the sample database, the pixel classes are encoded as numbers (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Segment:** Seven categories of outdoor photos were included in the database, from which the samples were randomly selected. Each image pixel was given a class by hand splitting the images.Each instance has a 3x3 size (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Sonar mines:** The dataset contains 111 patterns produced by reflecting sonar sounds off a metal cylinder at different angles and under other circumstances. In addition, there are 97 patterns found in rocks collected under identical circumstances in the file "sonar.rocks." A rising-frequency frequency-modulated chirp characterizes the transmitted sonar sound. Signals collected from various aspect angles, ranging from 90 degrees for the cylinder to 180 degrees for the rock, are included in the data set. Each pattern consists of a collection of 60 integers between 0.0 and 1.0. Each value is

an integration of the energy inside a specific frequency range over a particular amount of time. Since higher frequencies are sent later during the chirp, the integration aperture for these frequencies occurs later (Kluettermann 2022, surveys 2013, Rayana 2016).

- **Spam base:** The idea of "spam" is broad and includes pornographic material, chain letters, get-money-quick scams, and promotions for goods and websites. This data set contains a collection of spam emails sent by spam reportersăand the postmaster. The 650 area code or word "George" are a marker of non-spam since the collection of non-spam emails comes from filed business and personal emails. These help create a unique spam filter. To create a general-purpose spam filter, one must blind such non-spam indications or get a vast collection of non-spam. Hence this data set has diverse attributes that help classify spam and non-spam emails extracted from the collected data (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Spectf_0:** A diagnosis of cardiac Single Proton Emission Computed Tomography (SPECT) pictures is described in the dataset. There are two categories for each patient: normal and abnormal. The 267 SPECT image sets (patients) in the database were analyzed to extract characteristics that condense the original SPECT pictures. Forty-four continuous feature patterns were produced for each patient, and classification criteria were established using these patterns. It contains 267 occurrences described by 45 qualities, and the predicted attribute is binary, reflecting the overall diagnosis. SPECTF is an excellent data set for testing ML systems due to the variety of features and instances (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Speech:** The speech dataset includes actual information from English language recordings. The outliers are represented by seven additional speakers with varied accents, whereas the regular class comprises people with an American accent. A cutting-edge feature in speaker recognition, the feature vector is the i-vector of the speech segment. The 400 dimensions of the

dataset make it the challenge in our evaluation with the most dimensions. There are 3,686 cases, with 1.65% anomalies (Kluettermann 2022, surveys 2013, Rayana 2016).

- **Steel plates fault:** The Semeion Research Center of Sciences of Communication is the source of this data. With six different sorts of potential faults, the research's first goal was to correctly categorize the types of surface flaws in stainless steel plates. The "Other" category was added as well. The input vector had 27 markers that roughly describe the geometric shape and contour of the fault. This work was given to Semeion by the Centro Sviluppo Materiali (Italy). The study article, thus, is not able to go into depth about the 27 indicators used as input vectors or the kinds of the six classes of flaws (Kluettermann 2022, surveys 2013, Rayana 2016).
- **Vehicle van:** The silhouette of different kinds of vehicles is collected by the Statlog (Statlog 2022). The models are trained on these silhouette data sets to recognise a van (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Vowels:** The Vowel is undocumented dataset from UCI. Without the help of speaker, eleven British English vowel steady state is recognised with lpc derived training set. Vowel 0 is used as target class (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Waveform_5000:** The dataset is wave data produced by a wave generator that makes waves in three different classifications. Each class is created by combining two of the three "basic" waves. Next, the waveform is described by 40 other characteristics, all of which involve noise. The final 19 attributes have a mean of 0 and variance of 1, all noise attributes (Kluettermann 2022, Delft 2022, Rayana 2016).
- **Wbc:** The Wisconsin-Breast Cancer (Diagnostics) dataset (WBC) is a classification dataset from the UCI machine learning repository that contains cytology characteristics for breast cancer patients. Cytology is a fine needle breast examination with nipple discharge smear, touch preparation, and many more. These breast cancer biopsy cytology characteristics can

predict breast cancer categories. The two types are benign and malignant. In this dataset, the malignant class is downsampled to 21 points, regarded as outliers, whereas the benign class's points are inliers (Kluettermann 2022, surveys 2013, Delft 2022).

- **Wbc non ret:** This data set is similar to the previous WBC data set but has some more characteristics, and the target class is different. Here, the target class is non-returning, which indicates if cancer can recur or not (Kluettermann 2022, Delft 2022).
- **Synthetic linear data set:** This is the data set simulated in python to study the performance of the model on a linearly separable data set (Van Rossum and Drake 2009).
- **Synthetic donut data set:** This is the data set simulated in python to study the performance of the model on a Donut shaped data set (Van Rossum and Drake 2009).

Eidesstattliche Versicherung

(Affidavit)

Rao, Nikitha
Name, Vorname
(surname, first name)

229337
Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

☒ Masterarbeit
(Master's thesis)

Titel
(Title)

Improving Autoencoder Ensemble Learning for Unsupervised Anomaly Detection

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 02.11.22
Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 02.11.22
Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**