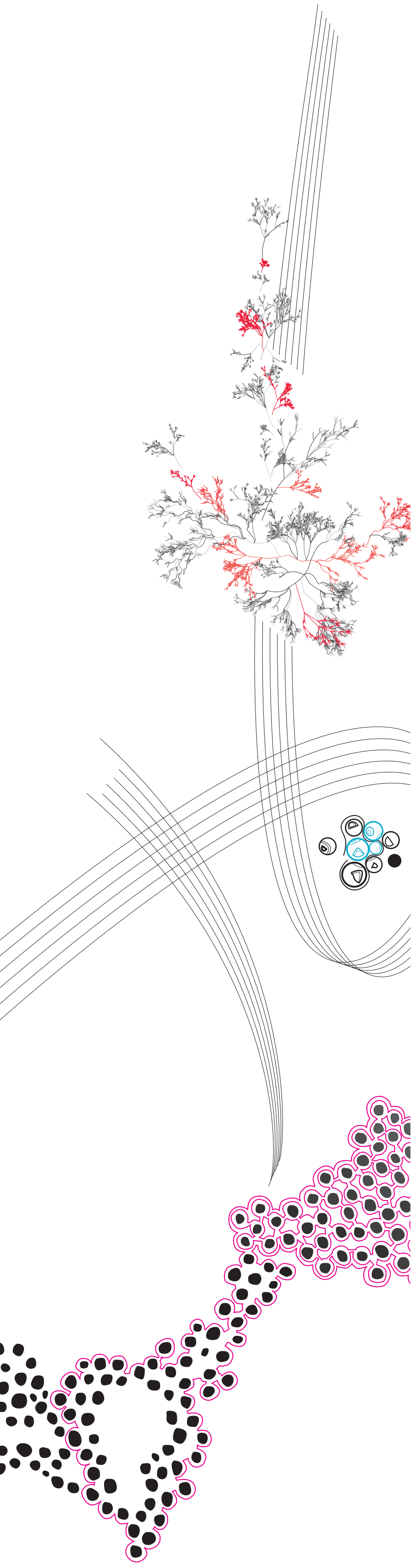
MSc Thesis Computer Science

Feature Extraction and Selection on Sparse, Complex, Sensor-Based Exhaled-Breath Data Sets



Bastiaan Frederik Mika van Tintelen

University of Twente Supervisors:
dr. N. Strisciuglio
n.strisciuglio@utwente.nl
dr.ing. F.W. Hahn
f.w.hahn@utwente.nl

The eNose Company Supervisors:
ir. H. Oord
henny.oord@enose.nl
dr.ir. J.W. Gerritsen
jan-willem.gerritsen@enose.nl

August 28, 2022

UNIVERSITY OF TWENTE.

the eNose company

Department of Data Management and Biometrics
Faculty of Electrical Engineering,
Mathematics and Computer Science

Acknowledgements

My graduation process has been a challenging but rewarding experience that could only be realized with the support of others.

First of all, I want to thank The eNose Company and all of its employees for receiving me with open arms, assisting me when needed, and providing feedback and insights. Especially I want to thank Marcel Bruins, André Elands, Jan-Willem Gerritsen, Bert Hoogendam, and Henny Oord.

From the University of Twente, I want to thank Nicola Strisciuglio for being my supervisor and providing guidance, suggesting noteworthy researches and providing feedback all while allowing me to find my own path through this process. Additionally, I want to thank Florian Hahn for being my second supervisor and providing insights from the viewpoint of a different research group.

I want to thank my fellow student and friend Ruben Lucas, who also conducted his Master Thesis at The eNose Company, for being my sparring partner during my Master Thesis and assisting me throughout my graduation process when needed.

Finally, I want to thank my friends and family for their support throughout my college years. Specifically I want to thank Alexander Stekelenburg, Mariëlle van Tintelen and Frits Tuininga.

Feature Extraction and Selection on Sparse, Complex, Sensor-Based Exhaled-Breath Data Sets

Bastiaan. F. M. van Tintelen
b.f.m.vantintelen@student.utwente.nl
s1866494

August 28, 2022

The aeoNose device, developed by The eNose Company, is used for diagnosing cancer by analysing the Volatile Organic Compounds residing within a person's exhaled breath. These Volatile Organic Compounds cause redox reactions at the surface of the device's sensors influencing their conductivity readings. These conductivity readings are then processed by peak shaving and either transforming or rescaling the signal data. This is followed by the Singular Value Decomposition to extract features by reducing the dimensionality of the data. Once all these preprocessing steps are applied the data is ready for disease classification with Machine Learning algorithms. In this paper, we propose an optimization approach for this data processing pipeline by first limiting the input data size as well as introducing the Natural Logarithmic rescaling function as preprocessing steps. This results in an area under the receiver operating characteristics curve increase from 59.29% to 66.69% on a 100 runs average for five classifiers. For this setup, the Extra Trees Classifier performs best with an average performance of 68.07%. Additionally, we exchanged the Singular Value Decomposition with a Fully Connected Autoencoder and a Convolutional Autoencoder resulting in a performance of 67.34% and 64.69% on average respectively. For both Autoencoders, the Random Forest Classifier performs best, resulting on average in 70.21% and 67.80% respectively. With this, we show that the Fully Connected Autoencoder can surpass the SVD compression with 2.14% when comparing best performing models, while the Convolutional Autoencoder can obtain similar results compared to the SVD compression.

Keywords: **Machine Learning, Autoencoder, Convolutional Autoencoder, Temporal Data**

1 Introduction

Exhaled-breath analysis is a non-invasive method for physical health management and is a promising development in the Health Care sector for early disease diagnosis and monitoring [1]. The eNose Company is one of the leading companies in screening diseases, such as colon cancer, lung cancer, and tuberculosis. This is realized by analysing the Volatile Organic Compounds (VOC) within exhaled breath. For this they have developed a dedicated point-of-care electronic nose [2] called the aeoNose (Figure 1) that measures these VOCs and records the conductivity readings of three sinusoidal-heated metal-oxide sensors. The conductivity of the sensors changes due to redox reactions occuring at the surface of these sensors between the VOCs and the sensor material.

The temperature of the sensor material influences the VOCs that attach to the surface of the sensor, therefore, resulting in different conductivity readings. This is realised by heating the sensor material between 260 °C and 340 °C regulated by two sinusoidal-waves [3]. These two sinusoidal-waves have a frequency of 0.15 and 0.075 Hertz resulting in a period of approximately 7 and 13 seconds respectively. Combined these two sinusoidal-waves form one data sampling cycle and have a duration of exactly 20 seconds.

During measurement of a patient's exhaled breath, the aeoNose is subjected to five different phases with unique air concentrations (Figure 2).

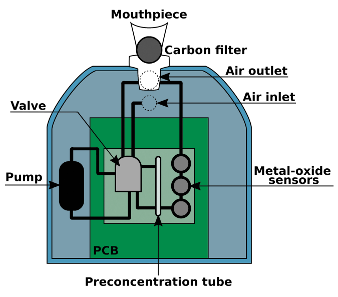


Figure 1: aeoNose Schematic [4]

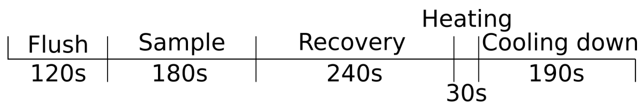


Figure 2: Sampling Timeline [4]

Phase one: Flush
During the Flush phase (120s / six cycles) the patients clean their lungs by breathing in through the mouthpiece via the carbon filter and out through the air outlet. The exhaled breath of the patient does not enter the device during this time, realized by locking the valve. Instead, clean air enters the device through the air inlet cleaning the preconcentration tube and metal-oxide sensors for measurement. Only a clean air data sample is collected during this phase.

Phase Two: Sample
During the Sample phase (180s / nine cycles) the valve opens and the patient's exhaled breath is pumped over the metal-oxide sensors causing the redox reactions between the sensor material and the VOCs. Additionally, the molecules within the airflow are stored in the preconcentration tube, after which it exits the device through the air outlet. This phase indicates the start of the exhaled-breath measurement and the conductivity readings are stored.

Phase Three: Recovery
During the Recovery phase (240s / 12 cycles) the valve closes again and the patient stops breathing through the device. Air enters the device through the air inlet and cleans the metal-oxide sensors of molecules while still recording conductivity readings.

Phase Four: Heating
During the Heating phase (30s / 1.5 cycles) the preconcentration tube is heated and starts letting go of stored molecules. The air enters the device through the air inlet, retrieving the molecules by passing through the preconcentration tube, and flows over the metal-oxide sensor for additional measurements.

Phase Five: Cooling Down
During the Cooling Down phase (190s / 9.5 cycles) the preconcentration tube starts cooling down while still letting go of stored molecules. The air still enters the device through the air inlet, retrieving the molecules by passing through the preconcentration tube, and flows over the metal-oxide sensor for additional measurements. After this phase, all data of the measurement have been collected.

For each sensor, the resulting data consists of 32 data sampling cycles as well as one cycle that contains the clean air data. In total, 32 conductivity readings are recorded for each sinusoidal-wave resulting in 64 conductivity readings for each data sampling cycle.

This data is then preprocessed to reduce the impact of possible sensor and/or analog-to-digital converter inaccuracies (resulting in incorrect conductivity readings), possible device and/or location-based differences (causing difficulties in training classifiers that perform well on data sets from different devices and locations), and data information that does not represent the screened disease (e.g. the sinusoidal-wave controlling the sensor temperature). On top of this, the data set is small and imbalanced as is common in the Health Care sector. This is caused by the limited number of participants during clinical studies of which only a minority have the screened disease. All these challenges influence the final classification accuracies and robustness of the trained Machine Learning (ML) models.

In order to improve the robustness, and possibly the accuracy, of the ML models we have investigated the Singular Value Decomposition (SVD) and compared it to the Fully Connected Autoencoder (FC-AE) and the Convolutional Autoencoder (CAE). Since the field of Data Science has been, and still is, rapidly evolving these state-of-the-art Deep Learning (DL) methods could provide new capabilities to better overcome the aforementioned challenges [5].

We first provide necessary background knowledge in section 2 before we show an in-depth analysis of the current data processing pipeline implemented by The eNose Company in section 3. This leads into an overview of the previously mentioned as well as other possible optimization approaches in section 4. The validation methodology used throughout the testings is discussed in section 5 and some preliminary testings have been set up in section 6. These preliminary testings provide a foundation for the testing and analysis of the proposed Autoencoder (AE) optimization approaches in section 7. Everything is finished off with the conclusion and further research in section 8.

The following Research Questions have been defined.

RQ1 **What is an improvement option for the current data processing pipeline implemented by The eNose Company?**

SRQ1.1 *What are the components of, and their impact on, the current processing pipeline?*

SRQ1.2 *What are the shortcomings of the current processing pipeline?*

SRQ1.3 *What are promising additions and changes to the current processing pipeline?*

RQ2 **What is the impact of the most promising additions and changes on the classification capabilities of the current data processing pipeline implemented by The eNose Company?**

SRQ2.1 What is a proper validation methodology for analysing the impact of additions and changes to the current processing pipeline?

SRQ2.2 What is the impact of the most promising preprocessing additions and changes?

SRQ2.3 What is the impact of the most promising feature extraction changes?

2 Background

We will explain some of the terminology and methods that are used throughout this paper as to provide the required knowledge to understand the applications fully.

Synthetic Minority Over-sampling Technique

A popular technique to overcome the challenges of an imbalanced data set is the Synthetic Minority Over-sampling Technique (SMOTE) [6]. SMOTE is used to augment data points from the under-represented class, which is the basic principle of oversampling, while keeping the over-represented class intact. This ensures that no observations are removed during the data set balancing. The unique oversampling approach applied with SMOTE is realized by creating synthetic observations instead of duplicating them. By multiplying the difference between the feature vectors of two data samples with a random number between 0 and 1 and adding this to one of the two feature vectors a new data sample can be created. These new data samples are created on the line segment between the two used feature vectors and forces the decision region of the minority class to become more general [7]. This can be translated to the following theorem [6, 329-330].
Let f_1 and f_2 be two data samples from the minority class. Then a new data sample can be generated for the minority class with:

$$s_1 = f_{1,2}' = f_1 + (rand(0,1) * (f_2 - f_1))$$

Singular Value Decomposition

One of the most popular tools in modern numerical analysis is the Singular Value Decomposition (SVD). Since the SVD rearranges and merges information that resides within the data by importance it can be used as feature extraction method. This rearranging and merging can be applied to real square matrices [8, p. 78], complex square matrices [9], and general rectangular matrices [10]. After feature extraction it is possible to apply feature selection by, starting from the most important column features, specifying the desired number of features. When the number of features of a data set are linearly separable this number can be significantly reduced by applying the SVD while retaining the majority of the information that resides within the data.

The rearranging and merging of information for square matrices (in the square brackets the method for rectangular matrices is given) is realized with the following theorem.

Let matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ be the data set consisting of m observations described by n variables and k_{mn} represent each measurement. Then there exists a square [rectangular] matrix $\mathbf{U} \in \mathbb{R}^{m \times m}[\mathbb{R}^{m \times k}]$, a rectangular [square] zero matrix with non-negative values on the diagonal $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}[\mathbb{R}^{k \times k}]$, and a square [rectangular] matrix $\mathbf{V} \in \mathbb{R}^{n \times n}[\mathbb{R}^{k \times n}]$ [where $k = min(m, n)$] such that:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

The columns of matrix $\mathbf{U}$ and the columns of matrix $\mathbf{V}$ are called left-singular vectors and right-singular vectors of $\mathbf{X}$ respectively. By taking $\hat{\mathbf{U}} = \hat{\mathbf{X}}\mathbf{\Sigma}^{-1}(\mathbf{V}^T)^{-1}$ the matrix $\hat{\mathbf{U}}$ can be determined for l new data observations $\hat{\mathbf{X}} \in \mathbb{R}^{l \times n}$. From this matrix $\hat{\mathbf{U}}$ the number of columns that corresponded to the desired complexity and amount of data can be taken, leaving a dimensionally reduced outcome compared to the original matrix $\hat{\mathbf{X}}$ with as much information aggregated as possible [11, p. 166].

3 Processing Pipeline Analysis

We provide an overview of the data processing pipeline used by The eNose Company to obtain a disease diagnosis from the raw exhaled-breath data as baseline (Figure 3).

3.1 Raw Data

The raw data contains the conductivity readings of the three sinusoidal-heated, metal-oxide sensors within the aeoNose device. For each measurement, this results in a total of 2048 conductivity readings for all three sensors where both sinusoidal-waves within a data sampling cycle have 32 conductivity readings resulting in 64 conductivity readings for each cycle. The conductivity readings of this data are temporally correlated as they are sampled directly after one another over time. Within these correlations some information might be hidden that could only be retrieved with specific processing steps. Additionally, one clean air data sample of 64

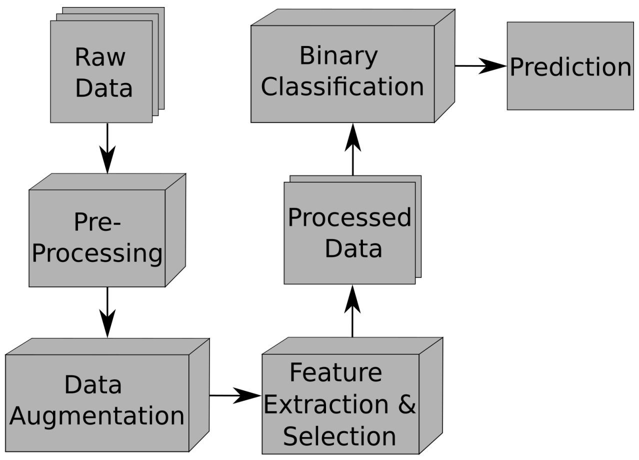


Figure 3: Data Processing Pipeline of The eNose Company

conductivity readings, subjected to the two sinusoidal-waves, is provided as some preprocessing steps require this information to transform the data.

3.2 Preprocessing

A plethora of preprocessing steps can be applied by The eNose Company as transformation options before feature extraction and selection with the SVD. These preprocessing steps comprise of peak shaving, signal transformation, rescaling, and sensor and cycle selection.

Peak Shaving

The metal-oxide sensors in the aeoNose device are slightly error-prone as they might output spikes that do not accurately represent the actual exhaled breath. The raw data should not contain these sudden spikes since redox reactions are chemical processes that should be represented in the conductivity readings. The analog-to-digital converter (ADC) could also cause spikes in the data due to incorrect transformation readings. Regardless of the origin of the spikes, the inaccuracies can negatively impact the final classification. To reduce the impact of these spikes peak shaving can be applied to the data by reducing the peaks to the average of the surrounding conductivity readings.

Signal Transformation

The sinusoidal-waves that are used to heat the sensors do not directly provide information about the person's exhaled breath. Therefore it might be beneficial to remove or alter this wave so the relevant information is not hidden. The sinusoidal-waves can be taken out of the data by subtracting the clean air from the data. Additionally, Cross-Correlation can be applied to the data by first transforming the data with the Fourier Transform [12, 13] followed by multiplying each of the 32 cycles with the complex conjugate of the clean air and back-transforming with the Inverse Fourier Transform. Cross-Correlation can also be applied to the two sinusoidal-waves in a cycle individually called Cross-Correlation subcycle. This ensures that the Fourier Transform is taken on only one specific sinusoidal-wave instead of taking both sinusoidal-waves in account at the same time. In all three cases, the sinusoidal-waves are either altered or removed and should no longer hide relevant information.

Rescaling

As an alternative to, or extension of, the signal transformation The eNose Company has implemented the natural logarithmic, exponential, square, and square root functions as data rescaling options. These rescaling functions transform the data and might therefore highlight more relevant information for feature extraction and classification.

Sensor and Cycle Selection

Since each of the three sensors of the aeoNose device have their own unique sensor material they gather different information due to the occurrence of different redox reactions. Selecting a subset, or combination, of sensors might therefore contain more relevant information. Additionally, since the sensors are exposed to different flows of air during the measurement selecting certain cycles might remove noisy and redundant information from the data set. Furthermore, this also reduces the amount of data within each data sample, slightly balancing the data discrepancy between the low number of data samples and the high amount of data within a data sample.

3.3 Data Augmentation

Data augmentation is used to increase the number of data samples for training the ML models. SMOTE is implemented to obtain a data set that has an equal number of positive and negative samples to ensure no class labels are significantly under- or over-represented compared to one another.

3.4 Feature Extraction and Selection

Dimensionality reduction is implemented with the SVD as feature extraction method in combination with feature selection. The number of conductivity readings, which depends on the number of cycles selected, is compressed to 17 features, which is an arbitrarily chosen number that complies with the limited participants size of clinical studies while retaining enough information for classification. The features outputted by the SVD are ranked based on their ability to represent the information within the original data. Of these features, the 17 with the highest explained variance are selected as representation of the information within the data.

3.5 Classification

As final step of the processing pipeline the preprocessed, compressed, and augmented raw data is fed to several ML models that return a classification of either diseased or healthy.

3.6 Processing Pipeline Shortcomings

Several shortcomings have been identified in the current processing pipeline that influences the overall classification performance of the ML methods. These shortcomings comprise of a limited data set, device- and/or location-based differences, feature extraction limitations, and non-deep classification.

Data Set Restrictions

Small and imbalanced data sets are quite familiar to the Health Care sector due to the restricted size of clinical studies and lack of testers that suffer from the screened disease. Ensuring the ML models are not overfitted on the training data is an ever-present challenge that needs to be overcome, which is significantly more difficult when dealing with small and imbalanced data sets.

Device- and/or Location-Based Differences

Even though The eNose Company has taken both hardware and software measures obtaining a generalized ML model that performs consistently on devices that are underrepresented or unseen during training remains a challenge. This is likely caused by differences in devices, such as slightly differently calibrated sensors, and/or locations, such as the ambient air in which the testing is conducted.

Feature Extraction Limitations

The SVD is a compression method that only considers the linear information within the data and therefore could lose valuable non-linear information. Investigating feature extraction methods that also consider this non-linear information might increase the performance of the trained ML models.

Furthermore, the lack of normalization or standardization might cause arbitrary large values to overrule lower values while the scaling of the data should not impact the information that resides within it. Additionally, the sensors of the aeoNose device will deteriorate over time which will result in different conductivity readings. Without normalization or standardization this might significantly impact the performance of the trained ML models. Adding normalization or standardization to the feature extraction might therefore contribute to a more robust ML model.

Non-Deep Classification

The ML methods that are currently implemented do not contain models from the Deep Learning (DL) subdomain. This is not necessarily a bad thing as there are situations where non-deep methods are preferred over DL methods, for instance when the amount of available data is limited [14]. This should indicate that DL methods will not be beneficial for this research since the available data set is small and imbalanced. However, the number of data for each paticipant in this study is quite high which could prove difficult for non-deep methods. Furthermore, the DL domain has been, and still is, rapidly developing and these newly obtained insights might provide new capabilities to the current processing pipeline.

4 Related Work

After analyzing the processing pipeline we have identified several possible solutions that can be applied to overcome some of the shortcomings. These possible solutions are finished with an in-depth explanation of AEs being the identified solution we focus on.

4.1 Possible Solutions

Possible solutions have been identified for the four shortcomings mentioned previously.

Data Set Restrictions Solutions

The first highlighted shortcoming is the restrictions on the data set, which is small in number of data samples but large in the number of conductivity readings for each data sample. The first part is somewhat tackled with the SMOTE data augmentation to obtain an equal prevalence between positive and negative data samples. However, SMOTE is not that suitable for significant increases in data samples since it is a linear augmentation method that might miss valuable information that resides in non-linear relation within the data. The second part is partly overcome by selecting stable parts of data samples in so called cycle subsets, but since the redox reaction is slow this will not reduce the number of conductivity readings significantly. A popular state-of-the-art data generation method is the Generative Adversarial Network (GAN) [15] [16] which can generate a large number of data samples that, given multiple layers are used, do take the non-linear information of the data into account. Additionally, GAN has shown great results in image processing, but it can be fine-tuned to be applied on temporal correlations [17]. This could therefore generate higher quality data samples compared to the SMOTE method and therefore it increase the classification quality.

Device- and/or Location-Based Difference Solutions

The second highlighted shortcoming is the differences between devices and/or locations that make it really difficult to obtain a generalized ML model that is consistent across devices and/or locations. One way to overcome this is to use Transfer Learning (TL) [18], in which a NN is trained on a source task and source domain that are closely related to the target task and target domain. Once this NN has been properly trained some layers can be locked from further training to retained learned patterns. The remaining part of the NN is then trained again on the target task with the target domain. The goal of this is to learn general information from the source domain while learning the specifics from the target domain. This is especially effective when the target domain has a limited amount of data.

Additionally, it is possible to use Supervised Contrastive (SupCon) learning [19] to overcome this shortcoming when finding a good source domain proves difficult. The source domain is now obtained by augmenting the target domain data. During training, the general features of the data set can be learned by only looking at the similarities and differences between data samples. This is realized by clustering data samples with the same targets while pulling the created target clusters apart. This splitting of the clusters could prove beneficial in overcoming the device- and/or location-based differences.

Feature Extraction Limitations Solutions

The third highlighted weakness is that the SVD feature extraction does not take into account non-linear information that might reside within the data. The first thing that might decrease the possible impact of this weakness is applying normalization between 0 and 1, normalization between -1 and 1, or standardization with mean 0 and standard deviation 1 before the SVD. This might both reduce the device differences as well as the deterioration of the sensors over time resulting in a more robust system. However, this still does not take into account the non-linear information that could reside within the data. To focus specifically on this non-linear information and the fact that the dimensionality of the data should be reduced feature extraction with AEs might be a suitable option. There are a lot of AEs available that can be applied to significantly reduce the dimensionality of the data in an unsupervised way while taking into account the non-linear information within a data set. Some AEs are designed specifically for temporally correlated sensor data as well [20].

Non-Deep Classification Solutions

The final highlighted shortcoming is the lack of classification testing with ML methods from the DL subdomain as these might provide new insights in the data. This is easiest tackled by implementing several Multilayer Perceptrons (MLP), Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN) for example. However, when the DL networks are preceded by the SVD the data is compressed based on linear separability which might hurt the performance of trained DL models. Preceding these methods with a feature extraction method that takes into account the non-linear relations will provide more information for the DL methods to work with.

4.2 Autoencoders

Based on the investigated shortcomings we propose to implement and analyze AE [21] as an alternative feature extraction method [22]. This possible solution has been chosen since it is early on in the processing pipeline and therefore impacts the effectiveness of solutions for data generation and NN classification. Focusing on the AEs before investigating these other shortcomings ensures that no re-investigation of these other implemented solutions will have to be conducted due to the impact the AEs might have on these implemented improvements. Additionally, AEs could also result in a more robust and more generally trained ML model due to their inherent ability to denoise the data. This could reduce the device and/or location-based differences that have been identified as well.

An AE is a NN that has the sole purpose of reconstructing the input data after mapping it to a lower dimension called the latent space as shown in Figure 4. This is realized by reducing the dimensionality of the input data x with an encoder g_ϕ to this latent space after which the latent space representation z is reconstructed back to the original dimensionality x' with a decoder f_θ. When the reconstructed input data x' is similar to the original input data x the AE has learned to map the input data to

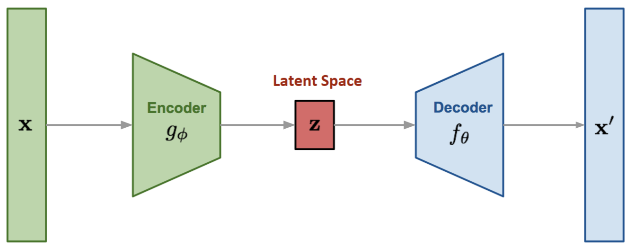


Figure 4: Autoencoder Architecture [23]

an "informative" representation of it in the latent space [21].

During training the parameters ϕ and θ of functions $g_\phi(x) = z$ $(\mathbb{R}^n \rightarrow \mathbb{R}^p)$ and $f_\theta(z) = x'$ $(\mathbb{R}^p \rightarrow \mathbb{R}^n)$ are updated by

$$arg\ min_{\phi,\theta}\ E[\Delta(\mathbf{x}, f_\theta(g_\phi(\mathbf{x}))][24]$$

where arg min indicates the parameter values for which the loss function is minimized, E the expected value over the distribution of **x**, and Δ the reconstruction loss function measured by the distance between the original input and reconstructed output. The most common loss function is the Mean Squared Error (MSE) loss (comprising both the E and Δ parts of the formula).

$$MSE(\phi, \theta) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - f_\theta(g_\phi(x^{(i)})))^2$$

The final combined function used for updating ϕ and θ is then defined as:

$$arg\ min_{\phi,\theta}\ \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - f_\theta(g_\phi(x^{(i)})))^2$$

In most AE the activation functions in g_ϕ and f_θ are non-linear [25], however, they could also be linear operations resulting in a linear AE [26], given that only one hidden layer is used. A linear AE achieves the same latent representation as the PCA, which can be calculated with the SVD, by also dropping the non-linear operations [27]. An AE can be seen as a generalization of the PCA where it learns a non-linear manifold instead of finding a low-dimensional hyperplane in which the data resides. [28, p. 1-2]). When both the AE and PCA are applied to linear data the difference is almost nonexistent, however, when applied to non-linear data the AE could significantly outperform the PCA when it comes to dimensionality reduction [22].

We have chosen to implement and analyze both the FC-AE and CAE since the FC-AE gives a good starting point comparison to the currently implemented SVD, since it is closely related to the PCA, and the CAE can take into account the temporal correlation within a data set which might prove beneficial.

Fully Connected Autoencoder

In the FC-AE both the encoder and decoder consist of fully-connected feedforward neural networks each containing one or more hidden dense layers.

For the encoder the input data x is directly, or via hidden layers, mapped to the latent space representation z and for the decoder this latent space representation z is directly, or via hidden layers, mapped to the reconstructed input x'.

[29] shows an application of (an ensemble of) three-layer FC-AEs on sensor data for human activity recognition. Their FC-AE approach obtains a better performance compared to other existing Human Activity Recognition methods.

Convolutional Autoencoder

In a CAE the encoder contains one or more convolutional layers before a dense layer is used to obtain the latent space representation and the decoder contains the transpose of these dense and convolutional layers to obtain the reconstructed input.

The encoder is applied to the input data x by applying one or more convolutional kernels to the data and mapping the data in the end to a latent space representation z with a dense layer. The decoder applies the transposed versions of the applied convolutional layers of the encoder in reverse order to the latent space representation z to obtain the reconstructed input x'.

[30] shows an application of CAEs for compressing and reconstructing Electroencephalogram Signals (EEG) restrained by a sparse and high-dimensional data set.

Additionally, [31] created a CAE that rectifies faults within the sensor data after first detecting faults with a Convolutional Neural Network (CNN), indicating a good (denoising) reconstruction capability of CAEs on sensor data.

5 Validation Methodology

We have set up a validation methodology that is used throughout all tests in this paper.

5.1 Validation Metrics and Methods

For analyzing the performances of the SVD, FC-AEs, and CAEs we used the Wilcoxon test in combination with the Area Under the Receiver Operating Characteristic Curve as well as test and data visualisation with t-distributed Stochastic Neighbor Embedding.

Wilcoxon

The Wilcoxon signed-rank test [32] is used to evaluate whether or not the difference in performance, when comparing two results, is statistically significant. Due to the non-parametric attribute it can be used to compare two populations when no underlying distribution has been established. This is important since no distribution has been determined for the sparse, complex, sensor-based data set we are working with.

Test set
A test set is used to apply the trained ML model to data that has not been seen during training. The test data set contains unseen data that has been collected with devices that have been taken into account during the training. This represents a real-life application scenario in which the trained ML model is used to classify new data samples. This test gives an indication whether or not a general model has been obtained for these devices.

Testing Restrictions
To create a realistic and balanced testing environment all conducted tests are subjected to some restrictions. First of all, the data set is split into 75% train and 25% test sets while ensuring that the number of positive and negative samples within a device are equal (Tables 1 and 2). This is realized by first balancing the test sets of each device to contain an equal number of positive and negative testing samples. Afterwards SMOTE is used to augment data samples to create a balanced training set for each device. An equal number of positive and negative samples across devices could not be realized due to the imbalance in data samples between devices. This is a difficult shortcoming to overcome since balancing with the current implemented options would have resulted in dropping a lot of real data from one device and/or generating a lot of data for another, both of which are not viable when used to an extreme. All tests also contain peak shaving to limit the impact of incorrect sensor readings as well as a data set containing only the conductivity readings of sensor B as The eNose Company previously obtained the best classification results with this setup for this specific disease. Subsequently, to ensure the impact of fluctuations between runs are kept to a minimum all setups have been run 100 times and the average over these runs has been taken as final result. Despite all this the trained AEs and ML classifiers still have a stochastic nature making it difficult to determine the exact impact a change has. Therefore all AUC scores and corresponding Wilcoxon test results are used as guidelines to find a suitable solution but they cannot be taken as facts. Finally to limit the run times the Extra Trees Classifier (ETC) [33] is used to test the performance of the setups. This specific classifier has been chosen as originally the best results were obtained with it on this data set. For a final overview we have decided to include four additional classifiers to get a better understanding of the impact of the setups comprising of Extreme gradient Boosting (XGBC) [34], Random Forest Classifier (RFC) [35], CatBoost [36] as well as a newly created MLP [37] to see the impact on a NN classifier.

t-Distributed Stochastic Neighbor Embedding
t-Distributed Stochastic Neighbor Embedding (t-SNE) [38] is a data visualisation tool used for representing high dimensional data in a 2D or 3D space. We use t-SNE to analyse the data distributions throughout the data processing pipeline as well as the impact of changes made to this data processing pipeline. We have decided to create t-SNE plots of the data with both devices and targets as labels since this gives more insight in the data than either t-SNE plot would give on its own.

Raw Data Samples			
Set	Pos	Neg	Total
Train	175	234	409
Test	59	78	137

Table 1: Raw Data Distribution

Data Samples with SMOTE			
Set	Pos	Neg	Total
Train	254	254	508
Test	76	76	152

Table 2: Data Distribution After SMOTE

6 Preliminary Testing

We have conducted a baseline analysis to get more insight into the data and the classification performance obtained with the SVD feature extraction. This baseline testing consists of data visualisation as well as a cycle and preprocessing steps analysis.

6.1 Data Visualisation

Before we started applying preprocessing steps to transform the data for classification we first created t-SNE plots of the raw data (Figure 5). These plots show that data points cluster for each device while the green and pink devices have some spread out outliers. This suggests device and/or location-based differences being present within the data. Additionally, the Positive and Negative targets are clustered together resulting in no clear separation between the two. Overall, the data points are clustered in such a way that finding a split between them will turn out to be challenging with the data as it is.

6.2 Baseline 0

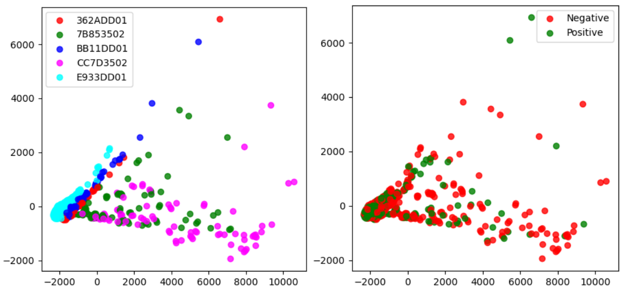
Baseline 0 is the initial processing pipeline used by The eNose Company and uses all 32 sinusoidal-wave cycles with the clean air removed with the Subtract method. This preprocessing setup obtains an ETC AUC score of 59.29% as average on 100 runs.

The t-SNE plots after all these preprocessing steps (Figure 6) show that the dense clusters of the raw data are spread out more resulting in one larger cluster. There are still device specific clusters within this cluster, but the data points are clearly more separated from one another. The Positive and Negative targets are also more spread out, however, a clear separation between the two targets is still lacking. Generally, the applied preprocessing steps are successful in creating a cluster where the data points are not mainly clustered around one specific point. t-SNE plots of all preprocessing steps for Baseline 0 can be found in Appendix A.1.

6.3 Baseline 1

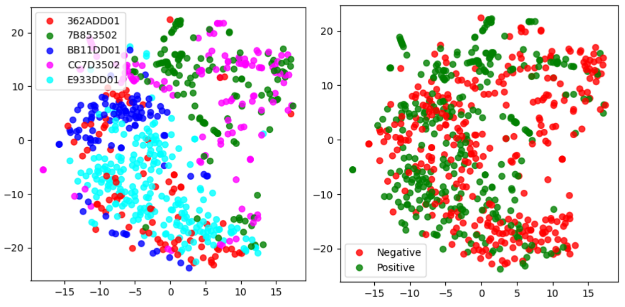
The first testing that we have conducted is a cycle analysis by investigating the neuron activations and ETC AUC scores of a convolutional encoder that is trained on all 32 cycles but tested on only one cycle. This test set is obtained by masking all cycles, but the one tested for, with zeros.

The analysis (Appendix B.1) shows that most useful information resides within cycles 10 to 25, which might be due to time needed for signal stabilization. This cycle selection perfectly aligns with the start of the Recovery phase of the Sampling, the Heating, and the first few cycles of the Cooling Down (see Figure 2 in section 1). Selecting these cycles seem to be effective since the ETC AUC score of 62.55% as average of 100 runs is, with an increase of 3.26%, statistically significant over Baseline 0 and serves as Baseline 1.



(a) Devices (b) Target

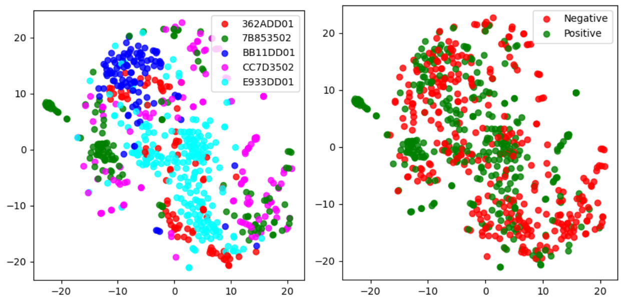
Figure 5: Raw Data T-SNE Visualisation



(a) Devices (b) Target

Figure 6: B0 Processed Data T-SNE Visualisation

Applying this cycle selection of cycles 10 to 25 show no clear difference in the t-SNE plots (Figure 7). The data points are still clustered together with some clear device clusters within it and there is still no clear separation between Positive and Negative targets. Nonetheless, the cycle selection proves effective since the AUC score increases statistically significantly. t-SNE plots of all preprocessing steps for Baseline 1 can be found in Appendix B.2.



(a) Devices (b) Target

Figure 7: B1 Processed Data T-SNE Visualisation

6.4 Baseline 2

Following the cycle analysis, we have tested several signal transformation and rescaling methods followed by normalization and standardization methods.

Signal Transformation Methods and Rescaling Functions

We have tested the Subtract method against the Cross-Correlation and Cross-Correlation sub-cycle signal transformation methods as well as the Natural Logarithmic, Exponential, Square, and Square Root rescaling functions (see Appendix C.1). The ETC performs best when the Natural Logarithmic rescaling function is used resulting in an AUC score of 68.07% as average on 100 runs. This setup performs statistically significantly better than Baseline 1 with an increase of 5.53%.

Normalization and Standardization Method

After exchanging the Subtract transformation method for the Natural Logarithmic rescaling function we tested all combinations of normalization between 0 and 1, normalization between -1 and 1, and standardization with mean 0 and standard deviation 1 both before and after the SVD feature extraction (see Appendix C.2). None of these combinations show a statistically significant improvement, or decline, with respect to the AUC scores over just applying the Natural Logarithmic rescaling function. This indicates that normalization and standardization methods are not a necessary addition to the processing pipeline, however, it is important to note that the application of these methods could significantly impact the performance when another data set is used, different processing steps are applied, or when the sensors of the device start deteriorating. Nonetheless, we will not apply normalization or standardization before or after the SVD, hence Baseline 2 will consist only of the Natural Logarithmic rescaling function instead of the Subtract transformation method.

t-SNE plot Analysis

The t-SNE plots (Figure 8 clearly show the effect of exchanging the Subtract transformation method with the Natural Logarithmic rescaling function as all devices, except red and pink, are split from each other. The fact that this split, which likely indicates the device and/or location-based differences, increases the AUC score so significantly feels counter-intuitive since a data set with no differences between devices is preferred. Within the device clusters there is still no clear separation between the Positive and Negative target. t-SNE plots of all preprocessing steps for Baseline 1 can be found in Appendix C.3.

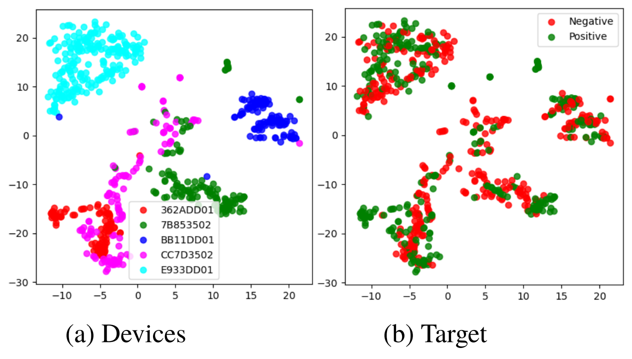


Figure 8: B2 Processed Data T-SNE Visualisation

6.5 Baseline Overview

With Baseline 2 established the preliminary testing has been concluded. The average performance for five ML classifiers on 100 runs as well as their combined average performance are shown as bar and AUC plots. Additionally, an AUC plot of the best performing classifier is provided as well (Figure 9). These three figures all show that Baseline 2 significantly outperforms both Baseline 0 and Baseline 1. Furthermore, Baseline 2 obtains on average roughly an increase of 9% compared to Baseline 0. This increase has been realized by selecting cycles 10 to 25 and exchanging the Subtract transformation method with the Natural Logarithmic rescaling function. The AUC scores goes up to 68.07% on a 100 run average for the ETC. The AUC plot shows, in both cases, that Baseline 2 has a superior average curve compared to Baseline 0 and Baseline 1.

7 Results

From the baseline testing in section 6 the analysis of the AEs have been set up. Besides the Testing Restriction mentioned in section 5 the AE testing also includes the cycle selection of cycles 10 to 25 and the Natural Logarithmic rescaling function as both result in a statistically significant increase in AUC scores.

7.1 FC-AE

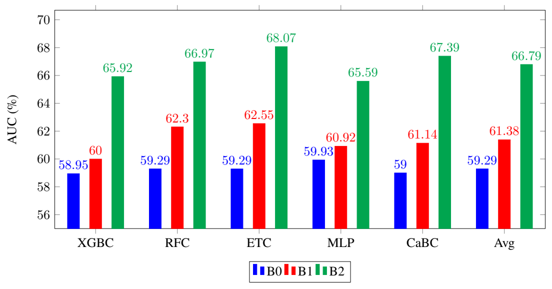
For the FC-AE we first provide the settings followed by the Network Testing and Normalization and Standardization Testing.

Settings

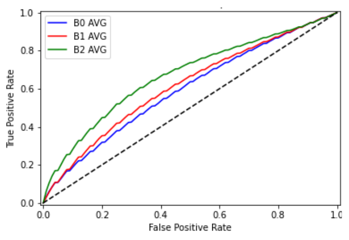
AEs have quite a number of parameters that can be tweaked during testing. To reduce the number of tweakable parameters we have fixed the parameters for the Latent Space, Optimizer, Loss Function, Activation Functions, Batch Size, and Epochs. This also ensures tests can be compared with one another since only one or a few parameters are changed between them.

Latent Space
We have chosen a latent space of 16 features on which the ML classifiers will be applied. This number has been chosen to remain close to the 17 features that the SVD is currently mapping to as to make comparing less complicated. The number 16 has been chosen above 17 to keep in line with the number of cycles and conductivity readings since all of them are powers of two.

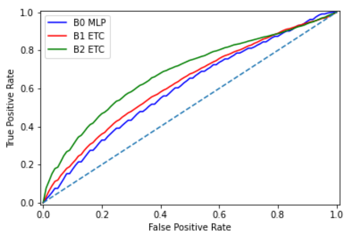
Optimizer
We have chosen the ADAM optimizer [39] as this is currently standard for training NNs.



(a) Bar Graph Comparison



(b) Average AUC Comparison



(c) Best AUC Comparison

Figure 9: Baseline 0 vs Baseline 1 vs Baseline 2

Loss Function
The MSE is used as the loss function since it is a straightforward method to determine how well the reconstructed output represents the original input since all values are directly compared to their reconstructed counterparts.

Activation Functions [40]
For all hidden layers, we have chosen to use ReLU activation functions [41] as this is standard for NNs. Since we want to classify on the latent space features a ReLU activation function in the output layer is not viable as it will result in roughly 50% of the features being zeros. Therefore we have chosen the sigmoid activation function to obtain a latent space between 0 and 1. For the reconstructed output, we have chosen a linear activation function to make training slightly easier since no values will be set to 0 when calculating the training loss.

Batch Size
Since the amount of data is limited we have chosen a batch size of 16 to keep it small as well as in line with the powers of two.

Epochs
Again as there is not a lot of data available we have chosen 50 epochs to keep it low to reduce the time needed for testing as early testing has not shown significant differences when increasing or decreasing this number.

Architecture Testing

We have conducted several tests with different architecture sizes investigating both one-layer and two-layer encoders and decoders, with the decoders being the inverse of the encoders. The in-depth results can be found in Appendix D.1. The testing shows that in this case the ETC obtains the best results with a two-layer encoder that maps the input data via a hidden layer of 128 neurons to an output layer of 16 neurons with an average AUC score after 100 runs of 66.36%. Even though this is the currently best performing setup it is still statistically significant worse than Baseline 2, meaning that it does not obtain a similar performance compared to the SVD feature extraction yet.

Normalization and Standardization Testing

We subjected the setup of a 128 neuron hidden layer followed by a 16 neuron output layer a Normalization and Standardization testing to investigate if, unlike the SVD, there is a statistically significant positive impact on the ML classification performance. The in-depth analysis of this can be found in Appendix D.2. Generally speaking, applying normalization between -1 and 1 and standardization with mean 0 and standard deviation 1 do not turn out to generate statistically significant better results over just the FC-AE. However, the results are comparable, indicating that they can be used as an alternative to just the FC-AE. The most important finding is that Normalization between 0 and 1 is always statistically significant over just the FC-AE independent of the normalization and standardization methods applied afterwards.

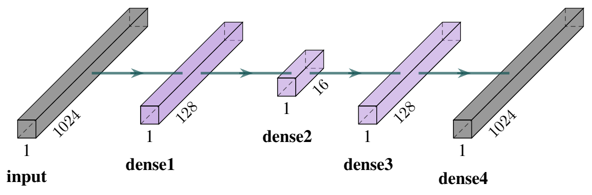
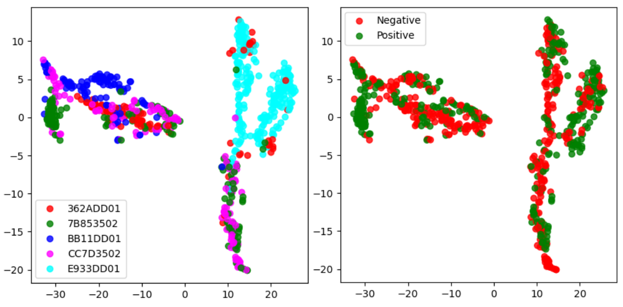


Figure 10: Architecture of the Fully Connected Autoencoder

Overview

The best performing FC-AE network setup is with a hidden layer of 128 neurons followed by an output layer of 16 neurons for the encoder (Figure 10). This setup in combination with Normalization between 0 and 1 beforehand yields an average of 100 runs AUC score of 68.62% for the ETC. This is slightly better than the 68.07% of Baseline 2 but not enough of an increase to be statistically significant. Nonetheless, a good-performing alternative to the SVD feature extraction methods has been found in the FC-AE. The FC-AE also has a clear impact on the t-SNE plot (Figure 11 since the four clusters of the SVD compression have been reduced to two clusters with data points from all devices, but the light blue devices, being present in both clusters. This could be an indication of a reduction in the impact of device and/or location-based difference, but with the current results that is hard to confirm. The FC-AE is, however, not able to find a clear separation in the Positive and Negative targets since they are still clustered together. t-SNE plots of all preprocessing steps for the FC-AE can be found in Appendix D.3.



(a) Devices (b) Target

Figure 11: FC-AE Processed Data t-SNE Visualisation

7.2 CAE

For the CAE, just like we did for the FC-AE, we first provide the settings followed by the Network Testing and Normalization and Standardization Testing.

Settings

All settings that have been defined for the FC-AE have also been used for the CAE.

- Latent Space of 16
- ADAM Optimizer
- MSE Loss Function
- ReLU Hidden Layer Activation Function, Sigmoid Encoder Output Activation Function, Linear Decoder Output Activation Function
- Batch Size of 16
- 50 Epochs

The CAE requires some additional settings compared to the FC-AE.

Stride
Since the latent space is mapped with a dense layer for classification it is important that the number of neurons that have to be mapped are minimized. As convolutional layers add filters the number of neurons increases significantly which makes the mapping to 16 features difficult. To ensure a mapping is viable we have decided to apply a Stride of 2 in every convolutional layer which reduces the number of values by half. In this way the dense layer has a less complicated task in obtaining 16 representative neurons in the latent space.

Pooling
After each convolutional layer a MaxPooling layer with stride 2 is added as is usual in Convolutional Networks. This layer simply considers two values at a time and returns the highest, resulting in a reduction of 50%.

Padding
We have chosen for "same" Padding to keep the number of values as a power of two.

Network Testing

We have conducted tests with several convolutional two-layer setups in combination with one dense layer of 16 neurons to get a latent space corresponding to both the SVD and FC-AE. In these convolutional layers, we tested several combinations of kernels and filters. The in-depth testing results are shown in Appendix E.1.

The first observation we made is that the number of filters should be limited to max 16 since all tests with more filters perform worse than their counterparts with fewer filters. The second observation is that, on the runs with 16 or less filters, the kernel size of three performs slightly worse compared to larger kernel sizes. On average the network with a convolutional layer of eight filters followed by a convolutional layer of 16 filters performs slightly more stable than both higher and lower kernel sizes. Furthermore, a convolutional layer with kernel size seven followed by a convolutional layer with kernel size five performs best across all setups with 16 filters or less. Ultimately a network consisting of a convolutional layer of eight filters with a kernel size of seven followed by a convolutional layer of 16 filters with a kernel size of five, obtaining an average AUC score on 100 runs of 64.18%, is chosen for further testing.

Normalization and Standardization Testing

Again we tested several normalization and standardization methods both before and after the CAE. The in-depth results for this can be found in Appendix E.2. The main observation here is that applying standardization between 0 and 1 yields a statistically significant improvement, with an average AUC score on 100 runs of 66.96%, over no normalization or standardization independent of a possible normalization or standardization applied afterwards. All other combinations of normalization and standardization both before and after the CAE yield similar results as no normalization or standardization.

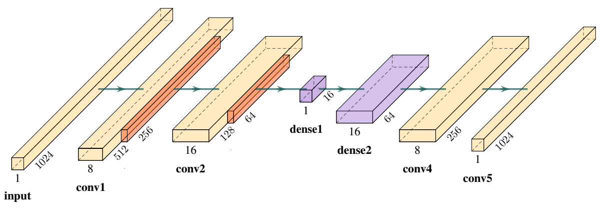
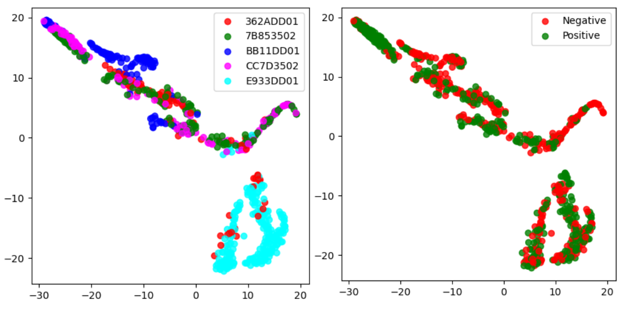


Figure 14: Architecture of the Convolutional Autoencoder

Overview

The best performing CAE is obtained when standardization between 0 and 1 is applied before a convolutional layer of eight filters with a kernel size of seven followed by a convolutional layer of 16 layers with a kernel size of five concluded with a 16 neurons dense layer (Figure 14) resulting in an average ETC AUC score after 100 runs of 66.96%. This is statistically significantly worse than Baseline 2 which is unexpected since the CAE usually performs well on temporally correlated data. A reason for this relatively bad performance might be that there is just not enough data available to properly train the convolutional layers and therefore limiting the learning capabilities of the CAE as a whole. The t-SNE plots (Figure E.3) show two clusters of which one primarily consists of the lightblue device while the other cluster has a no clear device clusters in it. Unfortunately, the Positive and Negative targets are still not separated in any clear way.



(a) Devices (b) Target

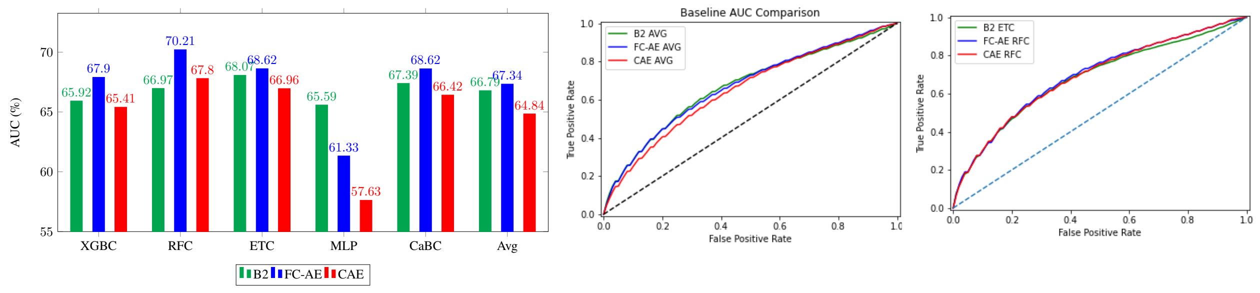
Figure 15: CAE Processed Data T-SNE Visualisation

7.3 Autoencoder versus Baseline Overview

We conclude the Baseline, FC-AE, and CAE testings with the average performance for five ML classifiers on 100 runs as well as their combined average performance as bar and AUC plots. Additionally, an AUC plot of the best performing classifier is provided (Figure 16). On average the FC-AE performs slightly better than Baseline 2, while the CAE performs significantly worse than both. This indicates that the FC-AE is a good alternative for the SVD compression. The CAE is lacking behind due to the poor MLP performance, which also negatively impacts the FC-AE average AUC score. When removing the MLP as classifier, the average AUC scores are 67.09%, 68.84%, and 66.46% respectively for Baseline 2, the FC-AE, and the CAE. This shows that the FC-AE performs even better, while also ensuring that the CAE is not lacking behind anymore. This indicates that the CAE is also a good alternative compared to the SVD compression given no MLP classifier is used. The second AUC plot also shows this by selecting the best performing classifiers. That is the ETC with 68.07% for Baseline 2, the RFC with 70.21% for the FC-AE, and the RFC with 67.8% for the CAE. The FC-AE outperforms Baseline 2 while the CAE is able to perform roughly equal to the SVD.

Now that we obtained an AUC score with the FC-AE that on average outperforms Baseline 2 we decided to investigate if it also increases the consistency per device individually (Appendix F.1). We have done this by training each setup on all data, but applying it only to the data points of one device. The main finding is that there are no real discrepancies between the three setups when it comes to the performance on individual devices. The maximum difference between AUC scores on a single device is roughly 5.5%, however this difference is mostly nullified with the performance on another device. This shows that the current implementations of the FC-AE and CAE do not obtain a more general feature representation compared to the SVD compression.

Additionally, we noticed that all three setups have a cluster that consists mainly of data points from the light blue device. Therefore, we have tested the three setups on a data set that does not contain these data points to investigate the impact on the data distribution (Appendix F.2). The AUC scores remain close to the AUC scores with the light blue device, while the data points are clustered closer together. This indicates that the addition of the light blue device does disrupt the clustering of the data, but does not significantly impact the AUC scores. Since this observation is the same for SVD as the AEs this will not be due to overfitting, but might indicate some restrictions within the data set.



(a) Bar Graph Comparison (b) Average AUC Comparison (c) Best AUC Comparison

Figure 16: Baseline 2 vs Fully Connected Autoencoder vs Convolutional Autoenecoder

8 Conclusions

The concluded tests show that the FC-AE with RFC can outperform the ETC of Baseline 2 statistically significant with 2.14%. Furthermore, the CAE can also obtain comparable results to the SVD compression as long as the MLP classifier is not used. This indicates that AEs can be effectively applied to this sparse, complex, sensor-based exhaled-breath data and obtain an average AUC score of 70.21% (for the FC-AE with RFC). We will conclude this research with a summary of all work related to the specified research questions as well as limitations and future works sections.

8.1 Research questions

For guiding purposes, we will summarize the conclusion by answering each Research Question in order.

What is an improvement option for the current data processing pipeline implemented by The eNose Company?

We found an answer to this question in section 3 by first investigating the current processing pipeline followed by shortcomings that lie within.

We identified that the available data set is restricted by its size and number of positive data samples due to the clinical nature in which the data is obtained. As a consequence, the ML methods are likely to overfit on the data without obtaining a general model that can be used in practise.

Additionally, the performance of the trained ML model differs greatly over devices, which indicates that some information resides within the data set that is specific to either devices or to the location in which the device is used. Identifying where this information resides within the data and countering it has not yet been effectively solved.

Furthermore, the SVD is used as feature extraction method which is limited to linearly separable data which is unsubstantiated for this specific data set. Moreover, the lack of normalization and standardization might introduce fluctuations in the performance of a trained model which is undesirable.

Finally, no ML classifiers that are part of the DL subdomain are currently being investigated. Given that this domain has been rapidly evolving in the last decade some new insight could be obtained with this.

For all these shortcomings we have listed improvement options ranging from data generation with the GAN to SupCon to RNNs and CNNs. However, we deemed AEs to be a suitable option for investigation since it impacts every mentioned shortcoming in one way or another.

What is the impact of the most promising additions and changes on the classification capabilities of the currently implemented processing pipeline?

We started answering this question by specifying a validation methodology in section 5 followed by an investigation of several preprocessing and feature extraction changes.

These changes have been translated to a baseline performance of 66.79% on average when limiting the data set to cycles 10 to 25 and exchanging the Subtract transformation method with the Natural Logarithmic rescaling function resulting in a total increase of roughly 9%. For this setup, the Extra Trees Classifier performs best with an average AUC score of 68.07%.

Afterwards, we have conducted several tests for the FC-AE and CAE. The FC-AE encoder has the best performance with a 128 neurons hidden layer and 16 neuron output layer resulting on average an AUC score of 67.34% which is slightly better than Baseline 2. The CAE encoder performance best with a convolutional hidden layer with eight filters and a kernel size of seven, followed by a convolutional hidden layer with 16 filters and a kernel size of five convolution hidden layer, and concluding with a 16 neuron output dense layer obtaining an AUC score of 64.69% on average. Indicating that AEs can be effectively used as an alternative to the SVD compression for this data set.

When comparing the best performing models of the SVD, FC-AE, and CAE it clearly shows that the RFC for the FC-AE is statistically significant over all other classifiers with an AUC score of 70.21%.

8.2 Future works

During this research several challenges have been encountered and solved, however, some challenges have not yet been tackled or have been revealed by the implemented solutions and investigating these might prove beneficial.

Firstly, we have shown that the AEs do not obtain a more general feature representation compared to the SVD compression which might be due to the limited data set. We would advise obtaining more data with additional clinical studies. When this is unobtainable we would suggest investigating more data augmentation and generation options like the GAN to overcome the problem of limited data.

Secondly, we have conducted an experiment investigating the performance of the SVD and AEs on individual devices which showed that the AUC scores are similar for the same individual device. This does not give a preference for once setup over another. Even though the AUC scores are comparable per device it does not directly indicate that the data points are classified similarly. We advice to perform an error analysis to investigate the performance of the SVD and AEs even more. If it turns out that they do not share correctly classified data points it might be worthwhile to investigate a multi-classifier system that combines multiple setups.

Thirdly, we would advise in-depth research into the combination of sensors since only sensor B is used during this research and this might currently limit the overall performance. Investigating ensemble learning might be a suitable option for this.

Fourthly, the current approach is only applicable to this specific data set and disease type which needs to be taken into account when classifying other data sets and diseases. A similar type of testing needs to be conducted to obtain a good-performing ML classification model that can be used in practise.

Finally, we were able to balance the data from each device but a balancing over all devices has not been obtained yet. An equal prevalence over devices might reduce the impact of device- and/or location-based differences since it is harder to overfit on one of the devices. However, the current imbalance in data point per device make it challenging to realize this.

References

[1] H. Tai, S. Wang, Z. Duan, and Y. Jiang, “Evolution of breath analysis based on humidity and gas sensors: Potential and challenges,” *Sensors and Actuators B: Chemical*, vol. 318, p. 128104, 2020.

[2] C. Baldini, L. Billeci, F. Sansone, R. Conte, C. Domenici, and A. Tonacci, “Electronic nose as a novel method for diagnosing cancer: A systematic review,” *Biosensors*, vol. 10, no. 8, 2020. [Online]. Available: https://www.mdpi.com/2079-6374/10/8/84

[3] M. Bruins, J. Gerritsen, W. Sande, A. van Belkum, and A. Bos, “Enabling a transferable calibration model for metal-oxide type electronic noses,” *Sensors and Actuators B: Chemical*, vol. 188, pp. 1187–1195, 11 2013.

[4] B. F. M. van Tintelen and R. H. Lucas, “Schemetic and timeline created for master theses on behalf of the enose company,” 2022.

[5] R. Wason, “Deep learning: Evolution and expansion,” *Cognitive Systems Research*, vol. 52, pp. 701–708, 2018.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[7] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, “Understanding data augmentation for classification: when to warp?” in *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–6.

[8] C. C. MacDuffee, “The theory of matrices,” -, 1933.

[9] L. Autonne, “Sur les groupes linéaires, réels et orthogonaux.” *Bulletin de la Société Mathématique de France, Tome 30*, pp. 121–134, 1902.

[10] C. Eckart and G. Young, “A principal axis transformation for non-hermitian matrices,” *Bulletin of the American Mathematical Society, vol. 45*, pp. 118–121, 1939.

[11] V. Klema and A. Laub, “The singular value decomposition: Its computation and some applications,” *IEEE Transactions on Automatic Control*, vol. 25, no. 2, pp. 164–176, 1980.

[12] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965. [Online]. Available: http://www.jstor.org/stable/2003354

[13] ——, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[14] P. Wang, E. Fan, and P. Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520302981

[15] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[16] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[17] N. Gao, H. Xue, W. Shao, S. Zhao, K. K. Qin, A. Prabowo, M. S. Rahaman, and F. D. Salim, "Generative adversarial networks for spatio-temporal data: A survey," *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 2, pp. 1–25, apr 2022. [Online]. Available: https://doi.org/10.1145

[18] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[19] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," 2020. [Online]. Available: https://arxiv.org/abs/2004.11362%7D

[20] N. Nguyen and B. Quanz, "Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting," 2021. [Online]. Available: https://arxiv.org/abs/2101.10460

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, p. 318–362.

[22] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016, roLoD: Robust Local Descriptors for Computer Vision 2014.

[23] L. Weng. From autoencoder to beta-vae. [Online]. Available: https://lilianweng.github.io/posts/2018-08-12-vae/

[24] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, ser. Proceedings of Machine Learning Research, I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, Eds., vol. 27. Bellevue, Washington, USA: PMLR, 02 Jul 2012, pp. 37–49. [Online]. Available: https://proceedings.mlr.press/v27/baldi12a.html

[25] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[26] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, pp. 53–58, 1989.

[27] E. Plaut, "From principal subspaces to principal components with linear autoencoders," -, 04 2018.

[28] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," -, 03 2020.

[29] K. Garcia, C. Rebelo de Sá, M. Poel, T. Carvalho, J. Moreira, J. Cardoso, A. de Carvalho, and J. Kok, "An ensemble of autonomous auto-encoders for human activity recognition," *Neurocomputing*, vol. 439, 01 2021.

[30] A. Al-Marridi, A. Mohamed, and A. Erbad, "Convolutional autoencoder approach for eeg compression and reconstruction in m-health systems," 06 2018, pp. 370–375.

[31] D. Jana, J. Patil, S. Herkal, S. Nagarajaiah, and L. Duenas-Osorio, "Cnn and convolutional autoencoder (cae) based real-time sensor fault detection, localization, and correction," *Mechanical Systems and Signal Processing*, vol. 169, p. 108723, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327021010414

[32] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945. [Online]. Available: http://www.jstor.org/stable/3001968

[33] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely Randomized Trees,” *Machine Learning*, vol. 36, pp. 3–42, 2006. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00341932

[34] ——, “Extremely Randomized Trees,” *Machine Learning*, vol. 36, pp. 3–42, 2006. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00341932

[35] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, 10 2001.

[36] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “Catboost: unbiased boosting with categorical features,” 2017. [Online]. Available: https://arxiv.org/abs/1706.09516

[37] M.-C. Popescu, V. Balas, L. Perescu-Popescu, and N. Mastorakis, “Multilayer perceptron and neural networks,” *WSEAS Transactions on Circuits and Systems*, vol. 8, 07 2009.

[38] L. van der Maaten and G. Hinton, “Viualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 11 2008.

[39] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

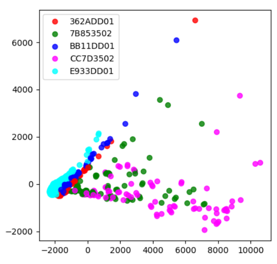
[40] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” 2018. [Online]. Available: https://arxiv.org/abs/1811.03378

[41] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML’10. Madison, WI, USA: Omnipress, 2010, p. 807–814.

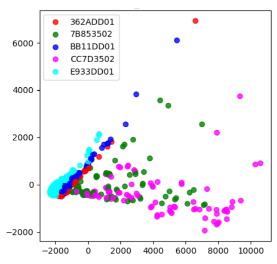
A Baseline 0

We provide the t-SNE plots of the entire processing pipeline for Baseline 0.

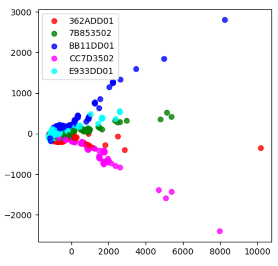
A.1 Data Visualisation



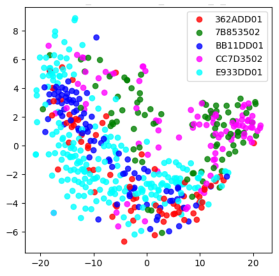
(a) Raw



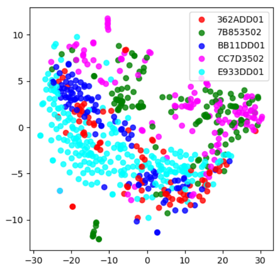
(b) Peakshaving



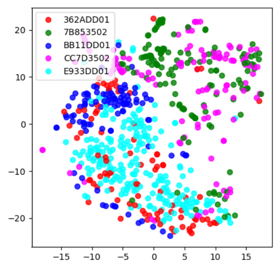
(c) Subtract



(d) Sensor B Selection

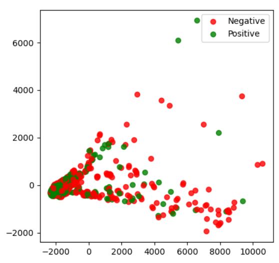


(e) SMOTE

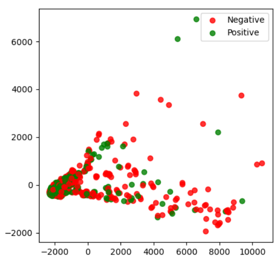


(f) Baseline 0 Processed

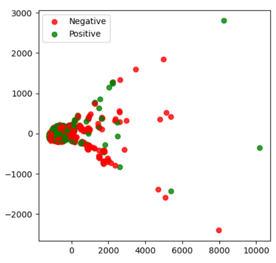
Figure 17: Data T-SNE Visualisation Devices



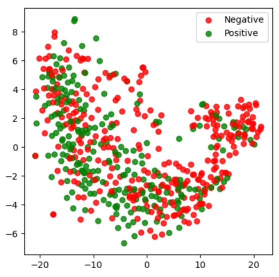
(a) Raw



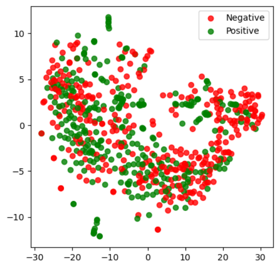
(b) Peakshaving



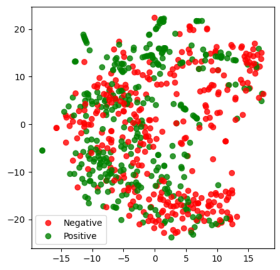
(c) Subtract



(d) Sensor B Selection



(e) SMOTE



(f) Baseline 0 Processed

Figure 18: Data T-SNE Visualisation Devices

B Baseline 1

We have conducted a cycle testing and provide t-SNE plots of the entire processing pipeline for Baseline 1.

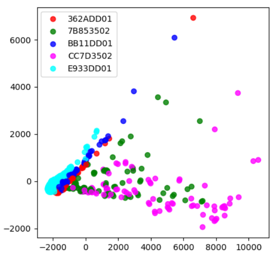
B.1 Cycle Testing

The first test we have conducted during preliminary testing is a cycle analysis (see Table 3). This is realized by masking cycles in the test set with zeros, e.g. when testing for the first cycle all values but those belonging to the first cycle are set to 0. We have trained a convolutional encoder on the train set as under normal circumstances. When subjecting the test set to the trained convolutional encoder we recorded the neuron activations and added them all together split per target as well as the ETC AUC scores. We did this 32 times, once for each cycle. Afterwards we calculated the differences between the positive and negative neuron activations. By highlighting all positive differences and ETC AUC scores above the 50% we get an indication where the relevant information resides within the data set. Eventhough there are some ETC AUC scores below the 50% it is still clear that the highest AUC scores are obtained from cycle 10 to 25-27. By selecting cycles 10 to 25, which is 16 of the 32 cycles, the data set has effectively been reduced by 50%. Comparing this to the Sampling phases shows that most information resides within the Recovery, Heating, and first few cycles of the Cooling Down phases.

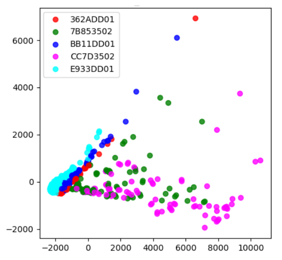
AUC and Wilcoxon Results: Cycles					
Cycle	Positive Neuron Activation	Negative Neuron Activation	Difference	ETC AUC	Sampling Phase
1	5562.96	5555.47	7.50	54.23	Sampling
2	5568.03	5569.73	-1.70	43.58	Sampling
3	5570.67	5575.35	-4.68	46.02	Sampling
4	5574.42	5579.78	-5.36	47.91	Sampling
5	5579.03	5583.18	-4.16	45.95	Sampling
6	5581.54	5585.19	-3.65	48.30	Sampling
7	5584.08	5586.81	-2.73	46.98	Sampling
8	5587.40	5588.28	-0.88	46.43	Sampling
9	5586.91	5586.57	0.33	46.37	Sampling
10	5578.50	5571.93	6.57	53.31	Recovery
11	5573.35	5554.16	19.19	52.81	Recovery
12	5565.55	5541.88	23.67	50.79	Recovery
13	5558.86	5541.88	23.00	52.99	Recovery
14	5556.07	5530.20	25.87	53.55	Recovery
15	5551.55	5524.72	26.83	55.07	Recovery
16	5549.78	5521.83	27.95	51.35	Recovery
17	5546.11	5519.59	26.52	55.34	Recovery
18	5544.51	5516.34	28.17	53.16	Recovery
19	5543.24	5515.66	27.58	48.92	Recovery
20	5542.20	5514.13	28.07	55.55	Recovery
21	5541.65	5513.51	28.14	54.83	Recovery
22	5541.09	5513.87	27.21	52.90	Heating
23	5543.87	5516.99	26.89	55.04	Cooling Down
24	5544.44	5526.84	17.60	52.23	Cooling Down
25	5546.93	5533.82	13.11	54.52	Cooling Down
26	5547.77	5535.49	12.28	49.60	Cooling Down
27	5545.61	5534.40	11.21	52.22	Cooling Down
28	5544.64	5531.89	12.75	49.69	Cooling Down
29	5543.15	5529.38	13.77	45.95	Cooling Down
30	5544.86	5527.25	17.61	45.75	Cooling Down
31	5542.15	5525.41	16.74	48.33	Cooling Down
32	5541.45	5523.73	17.72	48.96	Cooling Down

Table 3: AUC and Wilcoxon Testing Results: Cycles

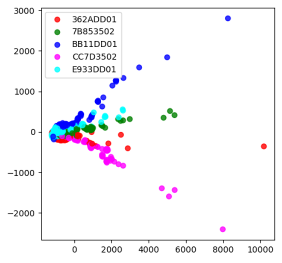
B.2 Data Visualisation



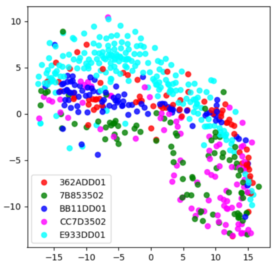
(a) Raw



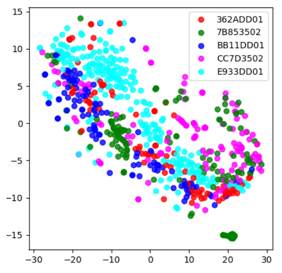
(b) Peakshaving



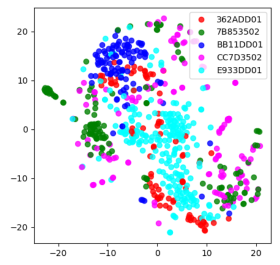
(c) Subtract



(d) Sensor B Cycle 10-25 Selection

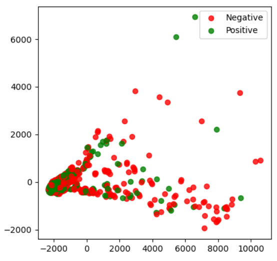


(e) SMOTE

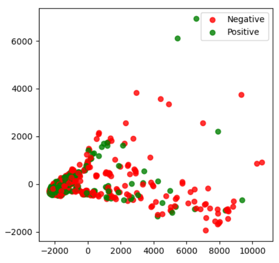


(f) Baseline 1 Processed

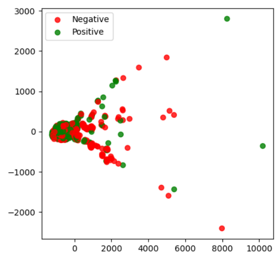
Figure 19: Data T-SNE Visualisation Devices



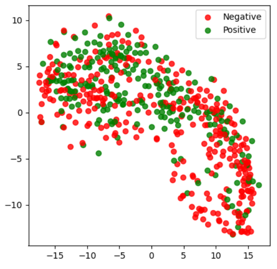
(a) Raw



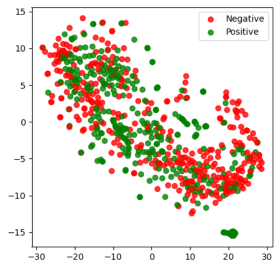
(b) Peakshaving



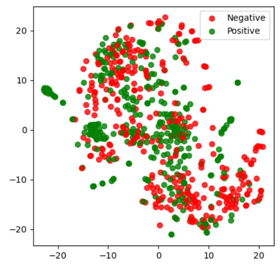
(c) Subtract



(d) Sensor B Cycle 10-25 Selection



(e) SMOTE



(f) Baseline 1 Processed

Figure 20: Data T-SNE Visualisation Devices

C Baseline 2

For Baseline 1 to Baseline 2 we have tested several Signal Transformation Methods and Rescaling Functions before the SVD as well as Normalization and Standardization methods both before and after the SVD. Additionally we provide the t-SNE plots of the entire processing pipeline for Baseline 2.

C.1 Signal Transformation Methods and Rescaling Functions

Table 4 shows the ETC AUC scores and Wilcoxon test scores when applying Signal Transformation Methods and Rescaling Functions before the SVD. All Wilcoxon test scores equal to or below 0.05 have been selected as statistically significant. A red text color indicates that the method in that row performs statistically significantly worse than the column method. A green text color indicates that the method in that row performs statistically significantly better than the column method. The Natural Logarithmic function clearly performs best as it is statistically significantly better than all other methods except the Square Root function.

AUC and Wilcoxon Results: Signal Transformation and Rescaling								
Method		Subtract	CC	CC-sc	Ln	Exp	Sq	Sqrt
	ETC	62.5466	66.4011	64.0793	68.0739	66.8090	65.8364	67.2604
Subtract	62.5466	x	0.00	0.00	0.00	0.00	0.00	0.00
CC	66.4011	0.00	x	0.00	0.00	0.86	0.36	0.02
CC-sc	64.0793	0.00	0.00	x	0.00	0.00	0.00	0.00
Ln	**68.0739**	0.00	0.00	0.00	x	0.00	0.00	0.10
Exp	66.8090	0.00	0.86	0.00	0.00	x	0.08	0.24
Sq	65.8364	0.00	0.36	0.00	0.00	0.08	x	0.01
Sqrt	67.2604	0.00	0.02	0.00	0.10	0.24	0.01	x

Table 4: AUC and Wilcoxon Testing Results: Signal Transformation and Rescaling

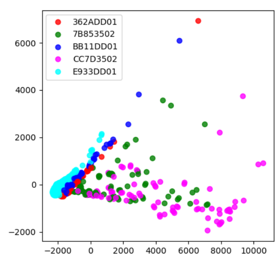
C.2 Normalization and Standardization

Table 5 shows the ETC AUC scores and Wilcoxon test values when applying Normalization or Standardization before and after the SVD compression. This is specific for a selection of cycles 10 to 25 and the Natural Logarithmic function for rescaling. All Wilcoxon test values equal to or below 0.05 have been selected as statistically significant. Each method is only compared to the SVD without Normalization and Standardization methods. The table shows that no combination of Normalization or Standardization has a statistically significant positive effect on the ETC AUC scores. Therefore the preferred setup is to just apply the SVD feature extraction without Normalization or Standardization. It is important to note that, although in this case it is not necessary, Normalization and Standardization afterwards is beneficial when another data set is used, different processing steps are applied, or when the sensors of the device are deteriorating.

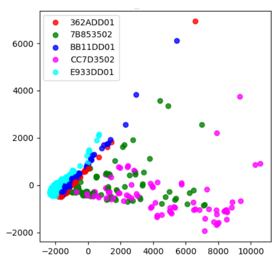
AUC and Wilcoxon Results: Normalization and Standardization		
Method	ETC	Wilcoxon
SVD	68.0739	x
SVD -> Norm 0 1	67.5236	0.22
SVD -> Norm -1 1	68.2200	0.75
SVD -> Stand mean std	67.9327	0.85
Norm 0 1 -> SVD	67.8820	0.55
Norm 0 1 -> SVD -> Norm 0 1	68.0828	0.69
Norm 0 1 -> SVD -> Norm -1 1	68.1497	0.95
Norm 0 1 -> SVD -> Stand mean std	68.0282	0.88
Norm -1 1 -> SVD	68.4676	0.36
Norm -1 1 -> SVD -> Norm 0 1	67.8500	0.54
Norm -1 1 -> SVD -> Norm -1 1	68.1059	0.97
Norm -1 1 -> SVD -> Stand mean std	67.9731	0.96
Stand mean std -> SVD	68.3309	0.53
Stand mean std -> SVD -> Norm 0 1	67.9265	0.68
Stand mean std -> SVD -> Norm -1 1	67.6546	0.27
Stand mean std -> SVD -> Stand mean std	67.9084	0.78

Table 5: AUC and Wilcoxon Testing Results: Normalization and Standardization

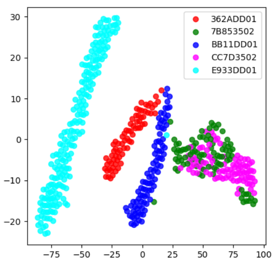
C.3 Data Visualisation



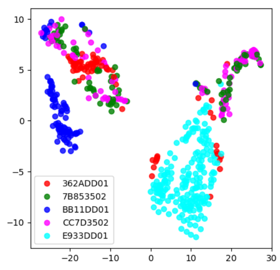
(a) Raw



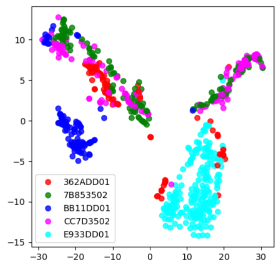
(b) Peakshaving



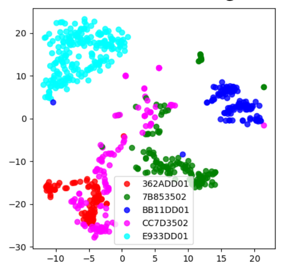
(c) Natural Logarithm



(d) Sensor B Cycle 10-25 Selection

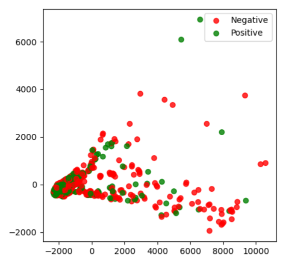


(e) SMOTE

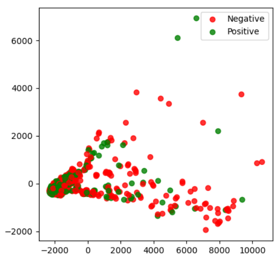


(f) Baseline 2 Processed

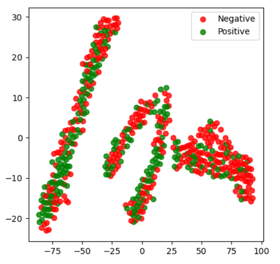
Figure 21: Data T-SNE Visualisation Devices



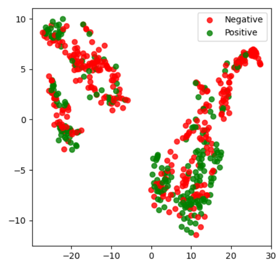
(a) Raw



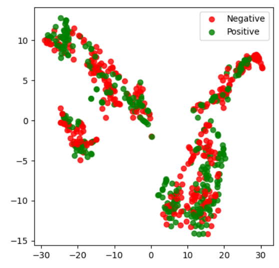
(b) Peakshaving



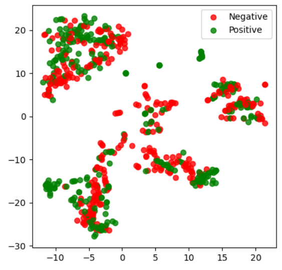
(c) Natural Logarithm



(d) Sensor B Cycle 10-25 Selection



(e) SMOTE



(f) Baseline 2 Processed

Figure 22: Data T-SNE Visualisation Devices

D FC-AE Testing

For the FC-AE we have conducted tests with respect to the architecture, Normalization and Standardization, and some additional testings. Additionally we provide the t-SNE plots of the entire processing pipeline for the FC-AE.

D.1 FC-AE Architecture Testing

We have tested one-layer and two-layer encoder architectures in which every layer outputs a number of neurons of the power of two. Table 6 shows the ETC AUC scores and Wilcoxon test values for these tests. This is specific for a selection of cycles 10 to 25 and the Natural Logarithmic function for rescaling. All Wilcoxon test values equal to or below 0.05 have been selected as statistically significant. Each architecture is compared to every other architecture with regard to the Wilcoxon test. A red text color indicates that the method in that row performs statistically significantly worse than the column method. A green text color indicates that the method in that row performs statistically significantly better than the column method. As the table shows there are three architectures that perform similarly (hidden layers with 256 neurons, 128 neurons, and 64 neurons) of which the architecture with hidden layer 128 neurons and output layer 16 neurons perform slightly, although not statistically significantly, better than the other two. We have chosen this architecture to conduct subsequent testings with.

AUC and Wilcoxon Results: Autoencoder Architecture Testing							
Layers		16	512 -> 16	256 -> 16	128 -> 16	64 -> 16	32 -> 16
	ETC	63.6767	62.6814	65.4970	66.3557	65.1033	63.5989
16	63.6767	x	0.20	0.00	0.00	0.01	0.72
512 -> 16	62.6814	0.20	x	0.00	0.00	0.00	0.29
256 -> 16	65.4970	0.00	0.00	x	0.10	0.76	0.00
128 -> 16	66.3557	0.00	0.00	0.10	x	0.07	0.00
64 -> 16	65.1033	0.01	0.00	0.76	0.07	x	0.02
32 -> 16	63.5989	0.72	0.29	0.00	0.00	0.02	x

Table 6: AUC and Wilcoxon Testing Results: Fully Connected Autoencoder Architecture

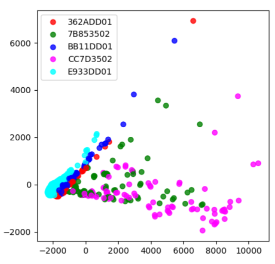
D.2 FC-AE Normalization and Standardization Testing

Table 7 shows the ETC AUC scores and Wilcoxon test values when applying Normalization or Standardization before and after the FC-AE. This is specific for a selection of cycles 10 to 25, the Natural Logarithmic function for rescaling and an Encoder with 128 neurons hidden layer and 16 neurons output layer. All Wilcoxon test values equal to or below 0.05 have been selected as statistically significant. Each method is only compared to the FC-AE without Normalization and Standardization methods. The table shows that Normalization and Standardization after the FC-AE and after the FC-AE with input data Normalized between 0 and 1 all obtain statistically significant improvements over just the FC-AE. Since only applying Normalization between 0 and 1 before the FC-AE obtains the best performance this is the preferred setup. It is important to note that, although in this case it is not necessary, Normalization and Standardization afterwards is beneficial when another data set is used, different processing steps are applied, or when the sensors of the device are deteriorating.

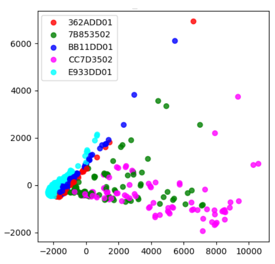
AUC and Wilcoxon Results: Normalization and Standardization				
Method			Test	
Before	AE	After	ETC	Wilcoxon
-	128 -> 16	-	66.3557	x
-	128 -> 16	Norm 0 1	67.7774	0.01
-	128 -> 16	Norm -1 1	67.4990	0.01
-	128 -> 16	Stand mean 0 std 1	67.9686	0.00
Norm 0 1	128 -> 16	-	68.6246	0.00
Norm 0 1	128 -> 16	Norm 0 1	67.9510	0.00
Norm 0 1	128 -> 16	Norm -1 1	68.1874	0.00
Norm 0 1	128 -> 16	Stand mean 0 std 1	68.3270	0.00
Norm -1 1	128 -> 16	-	67.3698	0.10
Norm -1 1	128 -> 16	Norm 0 1	67.1660	0.05
Norm -1 1	128 -> 16	Norm -1 1	67.0126	0.09
Norm -1 1	128 -> 16	Stand mean 0 std 1	66.4216	0.84
Stand mean 0 std 1	128 -> 16	-	66.6600	0.55
Stand mean 0 std 1	128 -> 16	Norm 0 1	67.2218	0.07
Stand mean 0 std 1	128 -> 16	Norm -1 1	65.5771	0.30
Stand mean 0 std 1	128 -> 16	Stand mean 0 std 1	65.5183	0.18

Table 7: AUC and Wilcoxon Testing Results: Normalization and Standardization

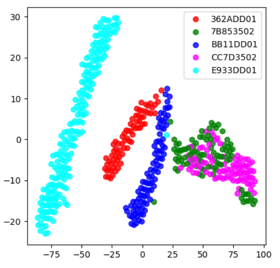
D.3 Data Visualisation



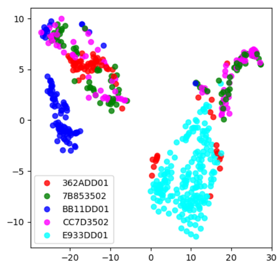
(a) Raw



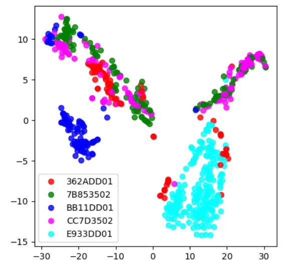
(b) Peakshaving



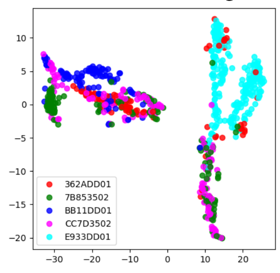
(c) Natural Logarithm



(d) Sensor B Cycle 10-25 Selection

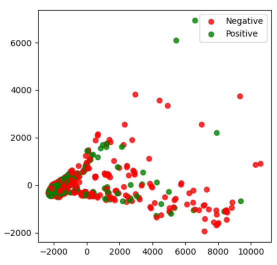


(e) SMOTE

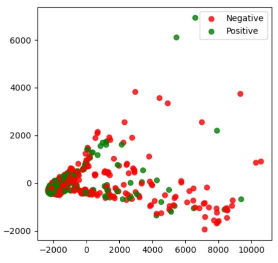


(f) FC-AE Processed

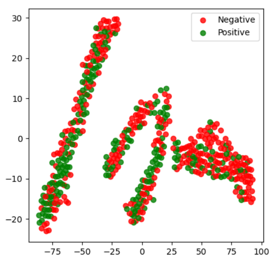
Figure 23: Data T-SNE Visualisation Devices



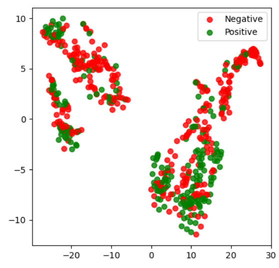
(a) Raw



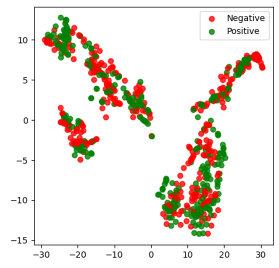
(b) Peakshaving



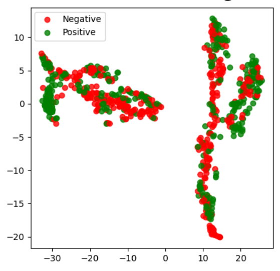
(c) Natural Logarithm



(d) Sensor B Cycle 10-25 Selection



(e) SMOTE



(f) FC-AE Processed

Figure 24: Data T-SNE Visualisation Devices

E CAE Testing

For the CAE we have conducted tests with respect to the architecture and applied Normalization and Standardization methods. Additionally we provide the t-SNE plots of the entire processing pipeline for the CAE.

E.1 CAE Architecture Testing

We have tested two-convolutional-layer encoder architectures of which every layer contains either four, eight, 16, or 32 filters and kernel sizes three, five, seven, or nine followed by a dense layer to 16 neurons. Table 8 shows the ETC AUC scores and Wilcoxon test values for these setups. This is specific for a selection of cycles 10 to 25 and the Natural Logarithmic function for rescaling. All Wilcoxon test values equal to or below 0.05 have been selected as statistically significant. Each architecture is compared to every other architecture with regard to the Wilcoxon test. A red text color indicates that the method in that row performs statistically significantly worse than the column method. A green text color indicates that the method in that row performs statistically significantly better than the column method. As the table shows the setups that contain a filter size of 32 perform generally the worst and are therefore not considered any longer. Of the remaining setups, both kernels 7 -> 5 obtain the best results. Since on average the 8 -> 16 filters have a more stable performance, we have chosen to continue with the kernels 7 -> 5 of this filter setup despite the fact that it slightly underperforms compared to the 4 -> 8 filter setup. The chosen architecture will be used for subsequent testings.

AUC and Wilcoxon Results: Convolutional Architecture											
Filters			4 ->8			8 ->16			16 ->32		
	Kernels		5 ->3	7 ->5	9 ->7	5 ->3	7 ->5	9 ->7	5 ->3	7 ->5	9 ->7
		ETC	62.7513	64.3739	64.0077	63.6447	64.1820	63.6248	62.2813	59.9729	58.3583
4 ->8	5 ->3	62.7513	x	0.00	0.05	0.11	0.01	0.10	0.49	0.00	0.00
	7 ->5	64.3739	0.00	x	0.52	0.33	0.53	0.17	0.00	0.00	0.00
	9 ->7	64.0077	0.05	0.52	x	0.94	0.54	0.60	0.01	0.00	0.00
8 ->16	5 ->3	63.6447	0.11	0.33	0.94	x	0.30	0.69	0.03	0.00	0.00
	7 ->5	64.1820	0.01	0.53	0.54	0.30	x	0.43	0.00	0.00	0.00
	9 ->7	63.6248	0.10	0.17	0.60	0.69	0.43	x	0.01	0.00	0.00
16 ->32	5 ->3	62.2813	0.49	0.00	0.01	0.03	0.00	0.01	x	0.00	0.00
	7 ->5	59.9729	0.00	0.00	0.00	0.00	0.00	0.00	0.00	x	0.04
	9 ->7	58.3583	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	x

Table 8: AUC and Wilcoxon Testing Results: Convolutional Autoencoder Architecture

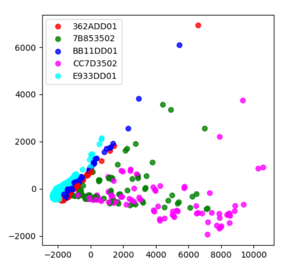
E.2 CAE Normalization and Standardization Testing

Table 9 shows the ETC AUC scores and Wilcoxon test values when applying Normalization or Standardization before and after the CAE. This is specific for a selection of cycles 10 to 25, the Natural Logarithmic function for rescaling and an Encoder with eight filters and seven kernels and 16 filters and five kernels convolutional hidden layers and 16 neurons dense output layer. All Wilcoxon test values equal to or below 0.05 have been selected as statistically significant. Each method is only compared to the CAE without Normalization and Standardization methods. The table shows that only Standardization with mean 0 and std 1 after the CAE obtains statistically significant improvement over just the CAE. Since only applying Standardization with mean 0 and std 1 before the CAE obtains the best performance this is the preferred setup. It is important to note that, although in this case it is not necessary, Normalization and Standardization afterwards is beneficial when another data set is used, different processing steps are applied, or when the sensors of the device are deteriorating.

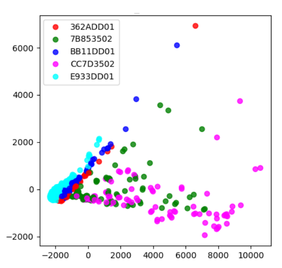
AUC and Wilcoxon Results: Normalization and Standardization				
Method			Test	
Before	AE	After	ETC	Wilcoxon
-	F: 8 -> 16 K: 7 -> 5	-	64.1820	x
-	F: 8 -> 16 K: 7 -> 5	Norm 0 1	64.3606	0.66
-	F: 8 -> 16 K: 7 -> 5	Norm -1 1	64.3547	0.69
-	F: 8 -> 16 K: 7 -> 5	Stand 0 1	63.3869	0.20
Norm 0 1	F: 8 -> 16 K: 7 -> 5	-	64.2690	0.77
Norm 0 1	F: 8 -> 16 K: 7 -> 5	Norm 0 1	64.8081	0.22
Norm 0 1	F: 8 -> 16 K: 7 -> 5	Norm -1 1	64.6061	0.50
Norm 0 1	F: 8 -> 16 K: 7 -> 5	Stand 0 1	64.9589	0.14
Norm -1 1	F: 8 -> 16 K: 7 -> 5	-	64.9092	0.26
Norm -1 1	F: 8 -> 16 K: 7 -> 5	Norm 0 1	65.3944	0.02
Norm -1 1	F: 8 -> 16 K: 7 -> 5	Norm -1 1	64.1938	0.76
Norm -1 1	F: 8 -> 16 K: 7 -> 5	Stand 0 1	64.4141	0.56
Stand 0 1	F: 8 -> 16 K: 7 -> 5	-	66.9559	0.00
Stand 0 1	F: 8 -> 16 K: 7 -> 5	Norm 0 1	66.9050	0.00
Stand 0 1	F: 8 -> 16 K: 7 -> 5	Norm -1 1	66.7844	0.00
Stand 0 1	F: 8 -> 16 K: 7 -> 5	Stand 0 1	66.6774	0.00

Table 9: AUC and Wilcoxon Testing Results: Normalization and Standardization

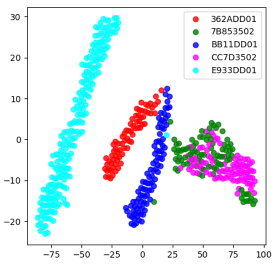
E.3 Data Visualisation



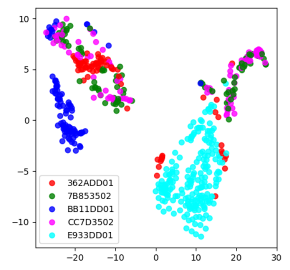
(a) Raw



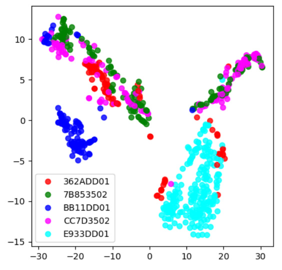
(b) Peakshaving



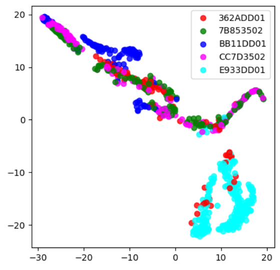
(c) Natural Logarithm



(d) Sensor B Cycle 10-25 Selection

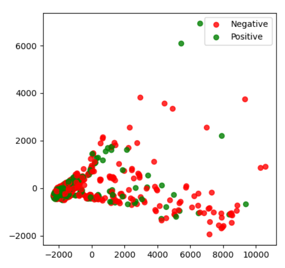


(e) SMOTE

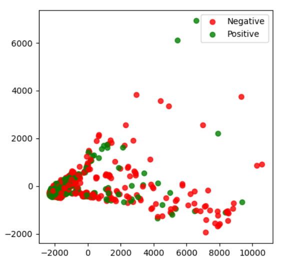


(f) CAE Processed

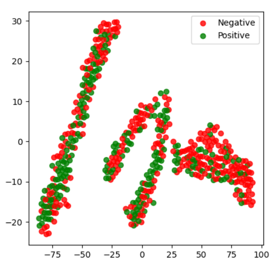
Figure 25: Data T-SNE Visualisation Devices



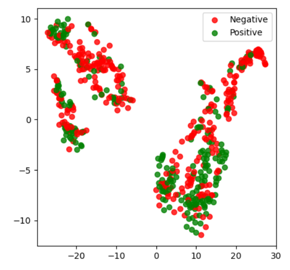
(a) Raw



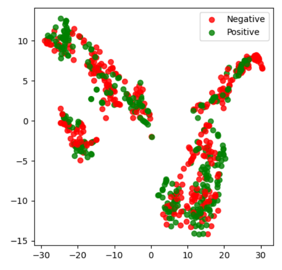
(b) Peakshaving



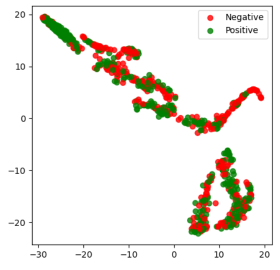
(c) Natural Logarithm



(d) Sensor B Cycle 10-25 Selection



(e) SMOTE



(f) CAE Processed

Figure 26: Data T-SNE Visualisation Devices

F Additional Experiments

We have conducted two additional experiments where we investigated the performance of the SVD and AEs on each device separately as well as an experiment where we left one device out of the data set to investigate its impact on the performance.

F.1 Device Performance Experiment

We have applied the SVD and AEs setups, train on all data, to each device individually to investigate consistency over devices. As Figure 27 shows there are no real consistency differences between the SVD, FC-AE, and CAE. The highest difference occurs in device BB11DD01 with roughly 5.5%, however this difference is nullified by device 7B853502 as the difference there is roughly 4% the other way. This indicates that the AE's do not obtain more general features compared to the SVD.

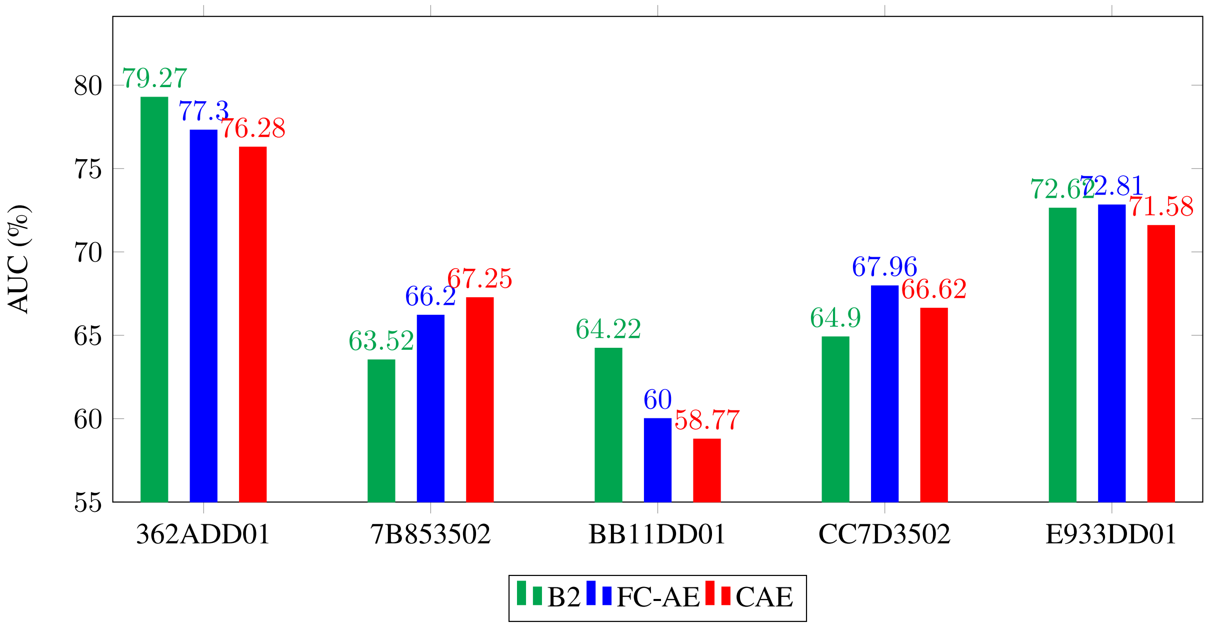


Figure 27: Average AUC Split Per Device

F.2 Device Removing Experiment

We have noticed in the t-SNE plot of the SVD, FC-AE and CAE that they all have a cluster consisting mainly of device E933DD01. By leaving this device out of the data set we investigate the impact it has on the AUC scores and the clustering of t-SNE plots. The AUC scores in Figure 28 are comparable to the AUC scores of Figure 27 that does contain device E933DD01. This indicates that the addition of this device does not significantly impact the performance of the SVD, FC-AE, and CAE. However, the impact of this device shows clearly in the t-SNE plots of Figure 29 since they are all one cluster in stead of at least two. Since this observation is the same for the SVD as the AEs the cause cannot lie in overfitting, but might indicate some restrictions within the data set.

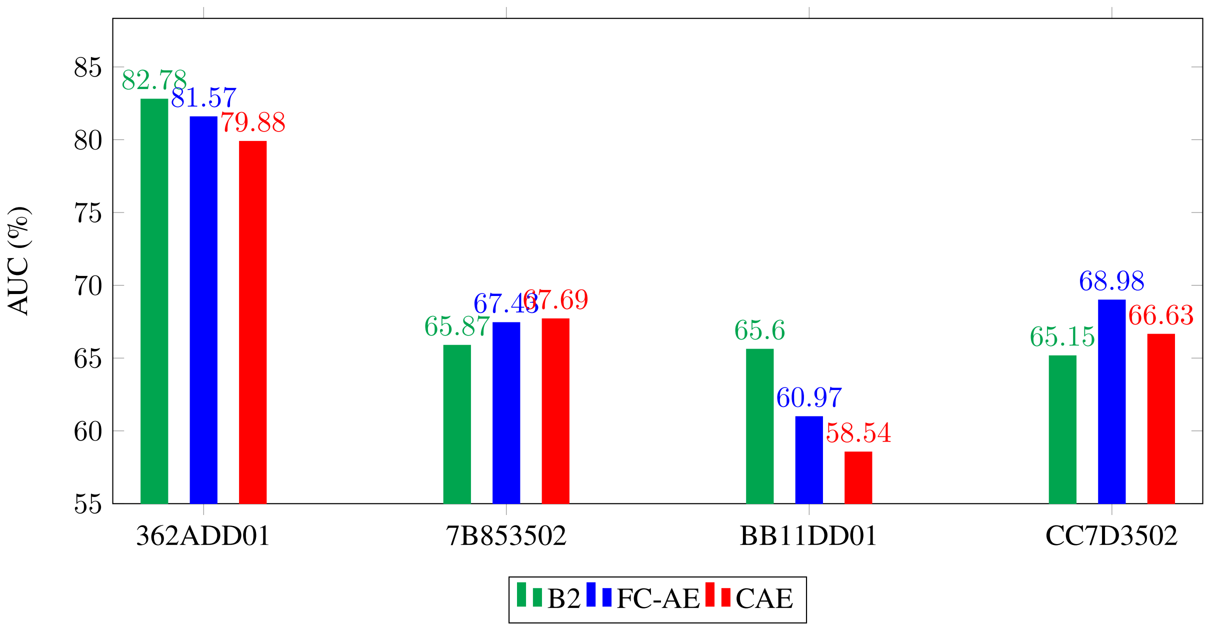
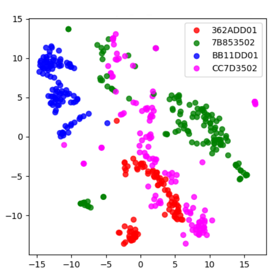
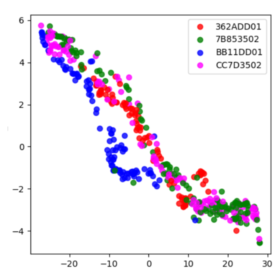


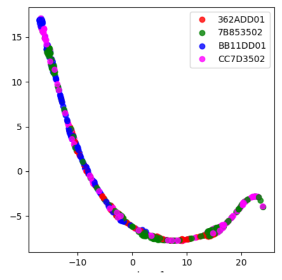
Figure 28: Average AUC Split Per Device



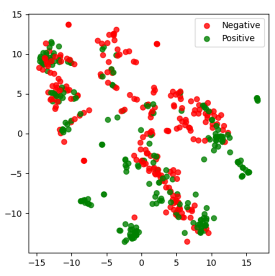
(a) SVD Device



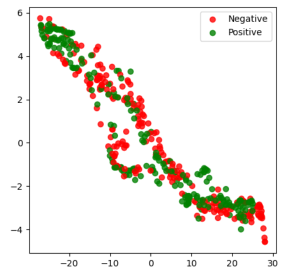
(b) FCAE Device



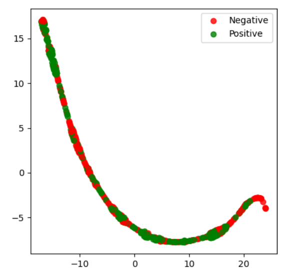
(c) CAE Device



(d) SVD Target



(e) FCAE Target



(f) CAE Target

Figure 29: Data T-SNE Visualisation Devices and Targets