UNIVERSITY OF TWENTE.

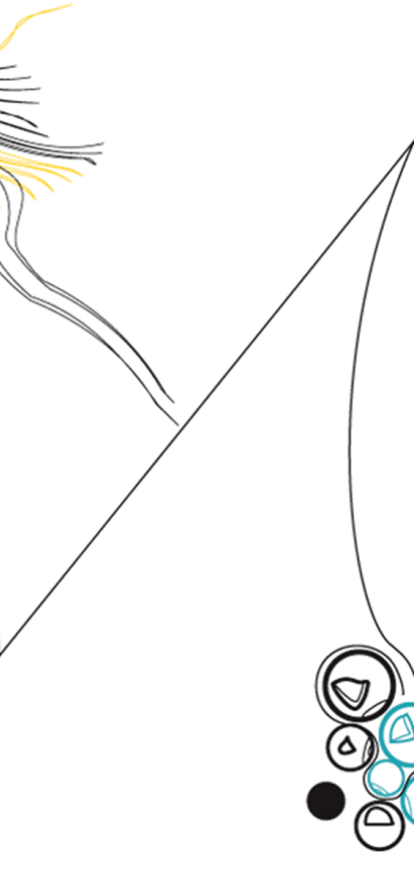
Master Thesis (201900200)

Improving Process Traceability Using Deep-Learning Based Unsupervised Feature Extraction

July 2020-Feb 2021

Prasanna Sosale Pavamana

March 14, 2021



Supervisor:
dr. Le Viet Duc

Faculty of Electrical Engineering
Mathematics and Computer Science
University of Twente
Zilverling 5013
P.O. Box 217
7500 AE Enschede
The Netherlands

Bosch Transmission Technology B V

Master Assignment

Improving Process Traceability Using Deep-Learning Based Unsupervised Feature Extraction

Prasanna Sosale Pavamana

Supervisor: dr. Nitin Bhushan

Bosch Transmission Technology B.V.

| Postbus 500 | 5000 AM Tilburg | NETHERLANDS |

Dr. Hub van Doorneweg 120 | 5026 RA Tilburg

Internal | TbP/TEF4 | 2020-09-16

Abstract

Computer Vision and Deep learning are making visible inroads in the manufacturing industry. One of the application areas in manufacturing includes process traceability. Process Traceability refers to the process of tracing parts back to the source components (e.g., raw material, machine, tools) that were used to produce the part. In Bosch transmission technologies, one component of a continuously variable transmission belt called as the elements are traced back to their source using manual methods based on measuring the part characteristics. The objective of this thesis is to automate this process. In particular, an investigation is conducted with deep learning algorithms to extract features in images of elements automatically, which are then used to achieve traceability. Experiments were performed to identify the best performing model in terms of traceability performance. Results indicate that the parts could be traced with only moderate success using a deep learning based automated method. However, there are definitive hints from the results suggesting that the performance could be improved with a better design on data collection.

Acknowledgements

First and foremost I am extremely grateful to my supervisors, Dr. Le Viet Duc and Dr. Nitin Bhushan for their invaluable advice, continuous support, and patience during my Master Thesis. Their immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank Prof. Paul Havinga and Dr. Luuk Spreeuwers for being part of my committee and providing their invaluable feedback over the course of the thesis. I would like to thank all the members in the TbP-TEF4 group in Bosch Tilburg. It is their kind help and support that made this study an enjoyable part of my life. Finally, I would like to express my gratitude to my parents for their continued support throughout. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study. Finally, I would also like to thank my friends who have helped me to stay motivated and focused during these challenging times.

Contents

List of Figures

List of Tables

Glossary

AAE Adversarial Auto-Encoders. 76

AE Auto-Encoders. 45, 75, 79

AOI Automatic Inspection Unit. 3–5, 15, 17, 18, 22, 23, 35, 59, 60, 67, 68, 83

ARI Adjusted Rand Index. 41

BRIEF Binary Robust Independent Elementary Features. 12

BRISK Binary Robust Invariant Scalable Keypoints. 12

CatGAN Categorical Generative Adversarial Network. 77, 79

CVT Continuously Variable Transmission. 3, 4, 13, 14, 23, 66

DC Deep Clustering. 40

DCC Deep Continuous Clustering. 74, 79

DCN Deep Clustering Network. 70, 71, 74, 79

DEC Deep Embedded Clustering. 2, 25, 31, 32, 54, 71, 74, 75, 79

DEN Deep Embedded Network. 71, 79

DEPICT Deep Embedded Regularized Clustering. 73, 74, 79

DMC Deep Multi-Manifold Clustering. 72, 73, 79

DNN Deep Neural Networks. 70, 75

DSC-Nets Deep Subspace Clustering Network. 71, 72, 79

ELBO Evidence Lower Bound. 75, 76

EOL End of Line. 6, 83, 85–88

FB Fine Blanking. 6, 15, 16, 18, 83, 85–88

FMI Fowlkes-Mallow Index. 41

GAN Generative Adversarial Network. 28, 31, 33, 77–79

GMM Gaussian Mixture Models. 1, 30, 40, 51, 54, 75, 76

GMVAE Gaussian Mixture Variational Auto-Encoder. 75, 76, 79

GPU Graphics Processing Unit. 21

HTML Hypertext Markup Language. 84

InfoGAN Information Maximizing Generative Adversarial Network. 77–79

JULE Joint Unsupervised Learning. 74

KL KullbackLeibler. 73

KLD KullbackLeibler Divergence. 27, 32, 40

MSE Mean-Squared Error. 2, 5, 7, 27, 37, 39, 40, 42, 45, 46, 55

OpenCV Open Source Computer Vision Library. 84

PCA Principal Component Analysis. 2, 26, 33, 36, 37, 39, 40, 45, 47, 48, 56, 58, 84, 87

RCC Robust Continuous Clustering. 74

ReLu Rectified Linear Unit. 13, 40, 71

RIM Regularized Information Maximization. 77

ROI Region of Interest. 5, 35, 36

SAE Stacked Auto-Encoder. 71

SDAE Stacked Denoising Auto-Encoder. 74

SGD Stochastic Gradient Descent. 74

SGVB Stochastic Gradient Variational Bayes. 75

SIFT Scale-Invariant Feature Transform. 12

SURF Speeded Up Robust Features. 12

t-SNE t-distributed Stochastic Neighbor Embedding. 84, 87

UI User-Interface. 84

VaDE Variational Deep Embedding. 75, 79

VAE Variational Auto-Encoder. 27, 31, 33, 75, 76, 79

Chapter 1

Introduction

1.1 Computer Vision and Deep Learning in Industry 4.0

Computer vision has made visible inroads into the manufacturing industry, especially with the ongoing industrial revolution (Industry 4.0). An example of how computer vision has revolutionized the printing industry is discussed in this paper [31]. A fully-automated visual inspection using AI to measure print quality is quoted to result in an accuracy of 98.4%. Similarly, industry 4.0 focuses on smart machines being made capable of diagnosing and analyzing issues without needing much or any human intervention. This approach plays a crucial role in overcoming traditional manufacturing overheads, such as human-induced errors, slow production or large-scale machine regression. Likewise, computer vision is already widely used throughout Bosch in many application areas including automated element inspection and defect monitoring systems.

Historically, one of the first instances of computer vision been applied in areas like inspection and process monitoring dates back to the 80s; see [22] and [33] for some of the early uses of computer vision in those application areas of manufacturing. Since then, the application areas of computer vision have expanded significantly. Some of the most recent ones include object detection, localization, and tracking [23] using deep learning. This research explores the possibility of utilizing the capability of computer vision and deep learning in improving the traceability in a manufacturing process.

For deep learning and traditional computer vision algorithms to function, there is a need to extract meaningful information from data because data may usually come in a large and unstructured format. One of the main differences between a traditional and

deep learning approach is the method of feature extraction.

Traditional computer vision involves handcrafting features through a process called Feature engineering. Feature engineering is a broader term that has feature selection and extraction as its two steps. Images usually have large dimensions and when unrolled, transforms into a large feature space. A large feature space is unrealistic to work with because of the required memory and processing time. Moreover, most of the information in an image is redundant, meaning they don't add much value to an intended task, but will still hamper the algorithm's performance. It is a general practice to use image feature descriptors to remove the redundant features. Some of the commonly used image feature descriptors are blobs, corners and edges. In addition, there are also more complex image feature descriptors such as, SIFT, SURF, BRISK and BRIEF; see [17] and [18] for a detailed survey of the different feature descriptors. The step where the right feature descriptors are selected based on the application needs is called Feature selection. When the relevant features are selected, the subsequent task is to extract those features from an image. This process is called Feature extraction. Figure 1.1 shows the steps of feature engineering. In an ideal supervised learning computer vision workflow, the features extracted will then be fed into a linear or logistic regression algorithm to perform a prediction or a classification task. For an unsupervised learning problem, where the task is to establish labels to data, these features may be applied on clustering algorithms, such as k-means and DBSCAN, to learn hidden patterns in data.

Figure 1.1: The feature engineering process

As discussed, traditional computer vision techniques are already widely used in industry 4.0 for industrial automation. But in recent years, deep learning has emerged as an efficient alternative for various application areas in manufacturing, especially in terms of feature engineering. OMahony et al.[24] discusses the advantages of deep learning over traditional computer vision. Deep learning is a sub-class of machine learning. In deep learning, the whole process of feature engineering is automated. That is, manual selection and extraction of features are no longer necessary. The speciality of a deep learning algorithm is automatic feature selection and extraction. The features are a result of statistical and probabilistic relationships among input variables.

For complex machine learning problems, such as pattern recognition and clustering, where feature selection plays an important role, manual feature selection is not a very efficient solution. Also, manually selected features that are not carefully handcrafted may affect the algorithm's accuracy. That is why deep learning outperforms traditional computer vision techniques.

Deep learning, is based on neural networks whose functioning is inspired by that of a human brain. A neural network consists of various neurons, each of which is capable of computing a specific result. A neural network consists of multiple layers, including an input layer, hidden layers, and an output layer. Each layer consists of several neurons, also called as units or perceptons. The input layer always has the same number of units as the number of input features. The output layer for a supervised classification problem has the same number of units as the number of classes. Neural networks also bring in non-linearity by using activation functions such as ReLu, Tanh and Sigmoid at each percepton. As discussed in [7], deep learning was observed to have better performance in object recognizing tasks as compared to hand-crafting features from high-dimensional data.

1.2 Background

1.2.1 Pushbelts

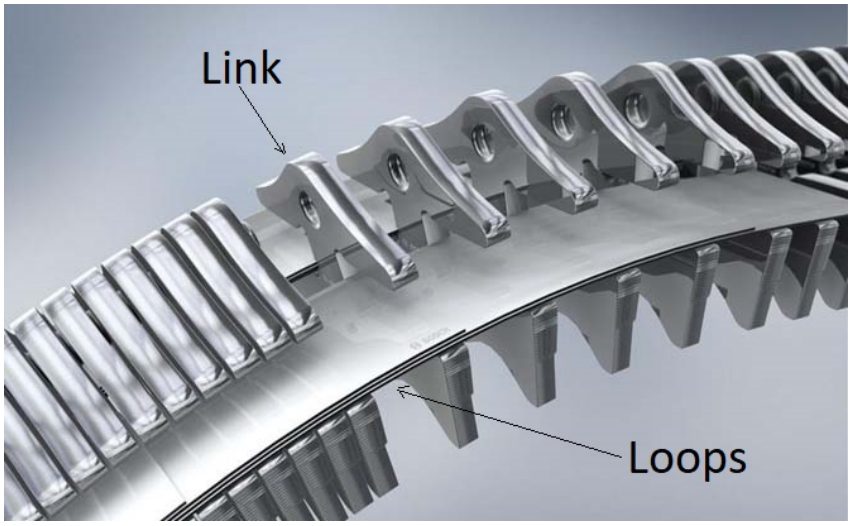
Bosch Transmission Technology produces pushbelts for automobiles with automatic gear-box systems. The pushbelts are an integral part of a Continuously Variable Transmission system (CVT) usually found in Automobiles. They run on a two pulley system, as shown in Figures 1.2. A detailed description of CVT is there in Appendix A. Pushbelts are made up of multiple steel objects called links and loops, as shown in figure 1.3. However, the focus of this research is on issues faced in the production of the links. The problem statement is introduced later in this chapter.

The next few sections introduces the process involved in producing the links. Figure 1.4 shows a representation of the actual production process for links.

(a) Pushbelt

(b) Continuously Variable Transmission

Figure 1.2: Bosch Product (a) Pushbelt - The product being manufactured in the Bosch Tilburg plant (b) The Continuously variable transmission (CVT) system where the Bosch pushbelt runs along two pulleys



(a) Pushbelt Components

(b) Link

Figure 1.3: Pushbelt components (a) The two metal components of Pushbelts called Loop and Link (b) The front side view of a Link

1.2.2 The Process

As seen in Figure 1.4, the raw material goes into a machine called as Fine Blanking (FB). The raw material used is a steel coil unrolled each time a new production run commences. A combination of tracks and tools present inside FB are responsible for stamping out the links from the raw material. The stamped out links get collected inside metal containers called Monos. The production happens in batches through multiple runs of the production cycle. Each batch results in monos with thousands of links. The monos then go through a series of intermediate processes starting from Heating before reaching a stage called Mixing. In this stage, the links from different monos are physically mixed, resulting in a Mix of links. The Mix is then distributed evenly among the same number of monos that went into Mixing. The Mix monos then go through the Automated Optical Inspection unit (AOI) where different regions of the links go through an optical inspection using cameras to identify defects.

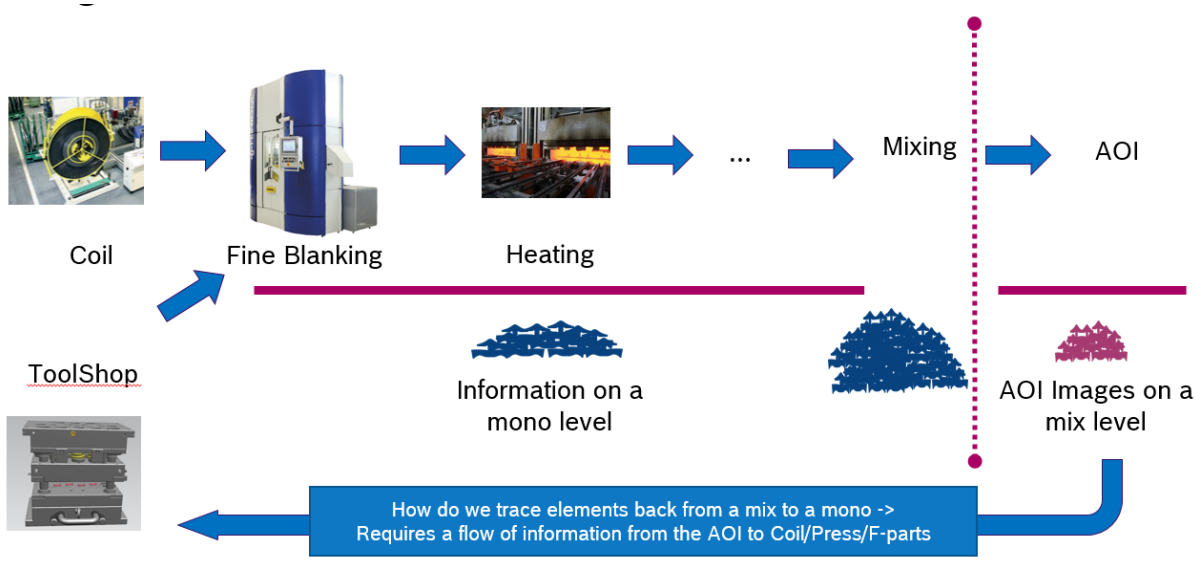


Figure 1.4: The production process used in the production of links starting from the raw material to the automatic optical inspection (AOI)

1.2.3 Fine Blanking

The links get produced by a technique called Fine blanking press (FB). In this technique, the links get stamped/pressed out of a raw steel strip using a combination of tracks and tools. Three tools used for this purpose are the cutter, the cutting plate and the ejector. The tools also are directly responsible for most of the resulting imprints on links. A single combination of the three tools is called a track. Each fine blanking machine will have four different tracks. During a regular run, all tracks function in parallel stamping out links at the same time. Links resulting from different tracks get collected

inside separate monos. In practice, the tools are refurbished after a certain number of runs to retain their sharpness. And for every run, each track produces around 21000 links. Hence, each FB machine at the end of a single run produces four monos with approximately 21000 * 4 links in total. Figure 1.5 shows a representation of the inside and the functioning of a fine blanking machine.

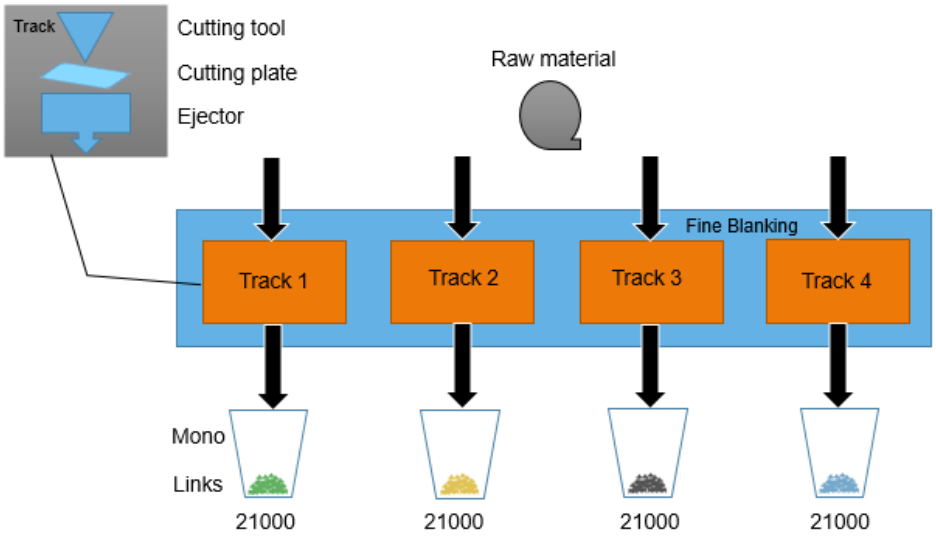


Figure 1.5: The components of a Fine Blanking machine comprising of a combination of tracks and tools

1.2.4 Mixing

The different monos produced through fine blanking go through a series of processes and then get eventually stored in an inventory. Although links from different monos don't differ in appearance and colour, there are significant characteristic differences such as the saddle surface height. However, these difference are only of a micro-scale that would be hard to visualize without a microscope. Yet, these differences have been observed to have an impact on the performance of the final product. Thus, mixing becomes necessary to smooth out the differences by physically mixing links from different monos. An elaborate mixing procedure is used to select the monos that gets sent into mixing.

Mixing is a process where a predefined number of monos get selected from the inventory, and the links in them are physically mixed. The number of monos that go to a mix usually varies between 9-10. Once a set of monos get chosen for mixing, the links of all these monos are physically mixed and distributed back into the same set of monos again. But this time each mono consists of links from the mix instead. The

major drawback of this process is that all the historical information that were previously available about the links get lost after mixing. Figure 1.6 shows the mixing process with the before and after mixing steps. All links do have the same colour, but at the before mixing stage, different colours are shown in the picture to indicate that each mono contains links belonging to only one process parameter (a combination of track and tools). At this stage, the labels such as mono ID and track ID are known for each link. In the after mixing stage, all the information is lost, and only the mix IDs are known for each link. Though using the mix ID it is still possible to obtain information about the monos that went into that mix, it is still very difficult to match the mono IDs to their links.

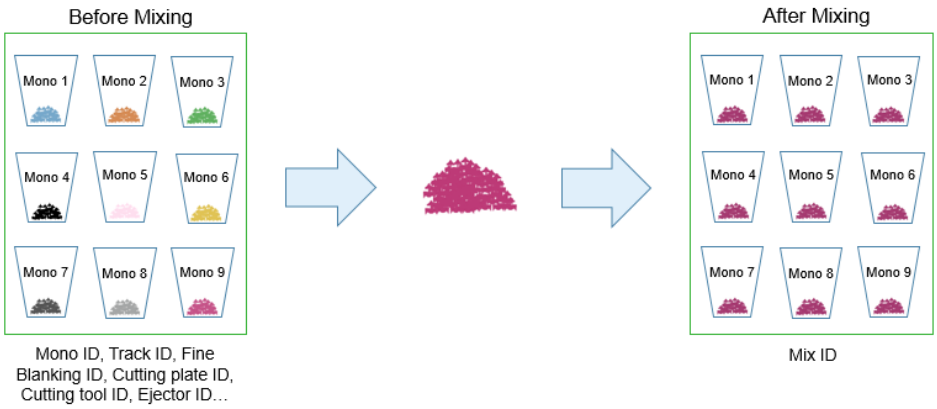


Figure 1.6: A representation of the process of Mixing where the links belonging to monos produced by different process parameters are physically mixed and distributed back into same monos again.

1.2.5 Automated Optical Inspection

Links in the mix monos then go through the Automated Optical Inspection Unit (AOI). The responsibility of AOI is to optically inspect links through cameras, and remove the defective pieces. As seen in Figure 1.7, in total, there are six stations in the AOI, each capable of performing an inspection on one specific region of a link. This research focuses on Station 6, which inspects the saddle surface of links. According to domain experts, this is where some most distinctive patterns that are unique to a combination of track and toolset could be found. Appendix A contains a detailed account of the AOI.

Figure 1.7: A representation of the Automated Optical Inspection unit (AOI) where the mixed links go through an optical inspections by 6 different stations

1.3 The Problem

The main objective of this research is to establish backward process traceability in the production lifecycle of links from the AOI to the toolset and FB, as seen in Figure 1.4. The research thus aims to establish mono labels to links whose source is no longer known after mixing. For this purpose, deep learning is used to extract features from images of the saddle surface in an unsupervised manner. Then clustering algorithms are used to group similar features. The final objective is to validate the meaning of the clusters obtained. The following section describes process traceability and its relevance to this research.

1.3.1 Process Traceability

Process traceability in manufacturing is the ability to trace all processes from raw material to all the way down the production line in both directions. In other words, it is the task of monitoring the movement of parts or whole products from the procurement of their raw material, till their assembly and delivery to the customer.

In this research's context, all the previous information such as the mono ID, FB ID, track ID are lost after the mixing process. Hence, there is a need to establish process traceability, to be able to trace a link backwards in the production chain. Given the critical role of the push belts in automatic gear transmission systems, there is a need to have a robust quality management system. A fault in the functioning of a push belt could not only prove to be dangerous to customers but can also be fatal. Thus it becomes extremely critical to establish backward process traceability to map the rejected links to their source. The tracing of rejected links to a particular track, run or FB machine could help Bosch to identify a possible pattern in the occurrence of rejects.

Knowing such patterns helps to take corrective measures to limit the number of rejects in future. For instance, after establishing traceability if it shows that the majority of rejects are due to track X, then a decision can be taken by Bosch to fix the issue in that track. From a business perspective, this digitization also helps Bosch to push further towards Industry 4.0. Also enables the organization to investigate the root causes for defects and consequently be profitable on cost, especially during the current uncertain market dynamics due to the COVID-19 pandemic.

Further, different strategies that could be employed to establish process traceability for links is discussed.

1.4 Process Traceability Strategies

Process traceability could be achieved using one of the following three strategies: manual traceability, semi-automated traceability and fully-automated traceability. Each of them will be described below, along with their advantages and drawbacks.

1.4.1 Manual traceability

Currently, Bosch has domain experts who follow manual work instructions to trace links back to their source. The links are physically inspected through a microscope to identify patterns based on the optical and geometrical characteristics of the links [6]. Hence, this method is purely an expert based traceability. In Appendix A, the entire work instruction is summarized.

Advantages

1. This method is a tried and tested one that is already being practised in Bosch and is purely based on domain expertise.

Drawbacks

1. As seen in Appendix A, there are many steps to be performed to decide for a single link. When these steps have to be repeated for 100s and 1000s of links, it would be time-consuming and remarkably inefficient.
2. As it is evident from the steps, a large portion of them is susceptible to human errors.

3. The expertise in this approach remains with the field expert. Once the expert decides to leave the company, the entire knowledge base is lost.

1.4.2 Semi-automated traceability

The semi-automated traceability is an approach where features are hand-crafted based on the domain expert's opinions. Here, along with the domain expert, an additional vision expert is involved. The vision expert works with digital data such as images and designs algorithms capable of extracting features suggested by the domain expert from images.Since the steps of feature selection and extraction are still expert dependent, this process is an expert assisted traceability

Advantages

1. Though, the features have to be hand-crafted, this method is more efficient than the manual approach because the features can be hand-crafted from a bulk of images at once.

2. Since the feature selection is still manual, there is control over the feature space.

Drawbacks

Though this method overcomes a few shortcomings from the manual strategy, there are still some drawbacks as listed below:

1. There is still a large dependency on the domain expert for feature selection.

2. Now that there are two people involved, a vision and a domain expert, it could be challenging if they do not understand their roles mutually.

1.4.3 Fully-automated traceability

The final strategy that could be used to achieve process traceability is a fully-automated one. Since there exist some challenges with those strategies discussed previously, this one can prove effective in overcoming them by providing a more robust and fully-automated solution. In brief, this approach uses deep learning to fully-automate feature extraction in an unsupervised manner.

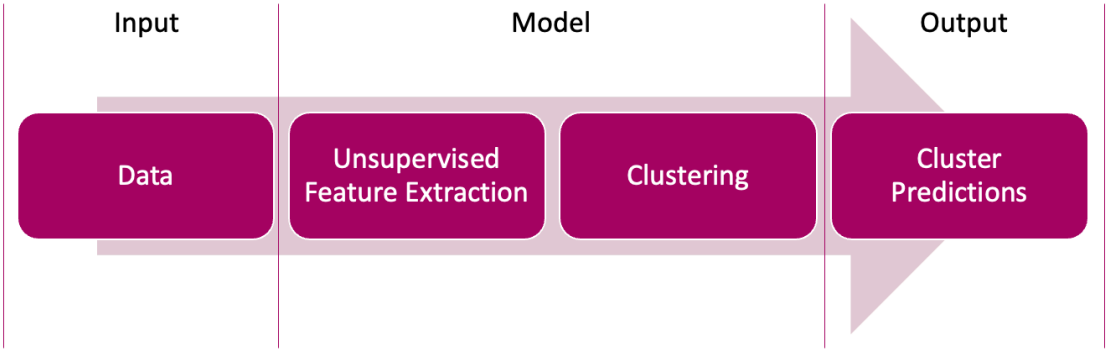


Figure 1.8: The Fully-automated pipeline with data as the input to the model. The output of the model is the cluster predictions

In Industry 4.0, one of the areas where deep learning could be made use of is in process traceability. The fully-automated approach uses deep learning for unsupervised feature extraction. Figure 1.8 shows a pipeline of a fully-automated approach where given data is first applied on any unsupervised feature extraction algorithms. The learnt features then go through a clustering algorithm that groups data based on feature similarity. The output of the pipeline is the predictions of cluster labels for each input datapoint.

Advantages

1. As this method is completely automated, it is the most efficient of them all. It can provide predictions for thousands of images in a short time.

Drawbacks

However, there are still some challenges, as discussed below:

1. Deep learning algorithms are usually data-hungry, meaning they need large amounts of data to learn patterns.
2. These algorithms also need a stable infrastructure with large memory and computation power such as GPUs to run on.
3. There is also no control over the features the algorithm could possibly learn.

1.5 Research Questions

How do we extract the features left by the process parameters on the links, and then trace them back to their source?

The aim of this research is to investigate the existence of a unique feature that can be then be traced back to their respective source. The source in this context is a process parameter such as track and tool combination. To address the main research question, following three sub research questions were formulated

RQ1: *Given unlabeled images of links, how do we optimally extract the features using Deep Learning?*

RQ2: *Given the set of extracted features, how to optimally cluster them into meaningful subgroups?*

RQ3: *How does the choice of the feature space affect the reconstruction and the clustering performance?*

RQ4: *Given that the clusters are detected, how can they be validated to establish their source?*

1.6 Limitations

Some of the practical limitations faced in carrying out the research are listed below.

Mix data collection

There were considerable challenges while collecting the mix dataset. Due to constraints in memory, The current AOI system in place is not capable of storing 200,000 images from an entire mix. Another major constraint was to not disrupt the normal production activities to collect the dataset. These two constraints made it possible to only collect images for the first 5160 links of the mix, which is less than 3% of the entire mix. Hence, the first assumption made was that this small sample of 5k odd images does represent the entire mix.

Mix data sampling and distribution

Though the images from first 5160 links from a mix are captured, the sampling method is still random as the links don't come in any particular order after mixing. Hence, the second assumption made was that these samples are uniformly distributed across the 9 monos in this mix. In the worse case, when the first two assumptions are wrong, then the data could have all samples only from coming from a single mono, hence resulting in a single cluster.

Validation data collection

The validation data or the mix monsters included 3 labelled samples from each of the 9 monos. While the images for mix data are collected automatically during the optical inspection, the images for the mix monsters have to be collected manually. Because the mix monster links are sampled before the element mixing is conducted and hence don't go through the optical inspection unit. That is also the reason why they preserve their labels, unlike the mix dataset. The procedure is to place each of these 27 samples on the station 6 of the AOI manually to click their images. Hence, there is some noise added at this step in terms of alignment, lighting and placement of the element. The image pre-processing step tries to minimize the influence of the alignment and lighting.

1.7 Thesis Outline

The remainder of this final report will be structured as follows: The next chapter, Chapter 2: Literature Survey, is a survey of the related works and state-of-the-art techniques used to solve similar feature extraction and clustering problems. In Chapter 3: The Methodology, the datasets will be introduced, along with the proposed method, model selection and evaluation approaches. In Chapter 4: Results, all the results will first be documented and later in Chapter 5: Discussion, the results will be critically discussed and analyzed, before finally concluding with the observations in Chapter 6, Conclusion. In the Recommendations chapter, various ways to improve the performances of the applied methods will be discussed. Finally, the Appendix includes all the additional information from the research that is not included in the main content of the report. The section includes an overview of CVT and the manufacturing process of the push-belts used in Bosch. In addition to the related works discussed in the next section, the Appendix also includes more methods from the literature that have been explored and relevant to this research. Finally, this report concludes with a description and results of the feasibility study conducted in the initial phases of this research. Figure 1.9 summarizes the thesis outline and displays the connection between different chapters in

the thesis.

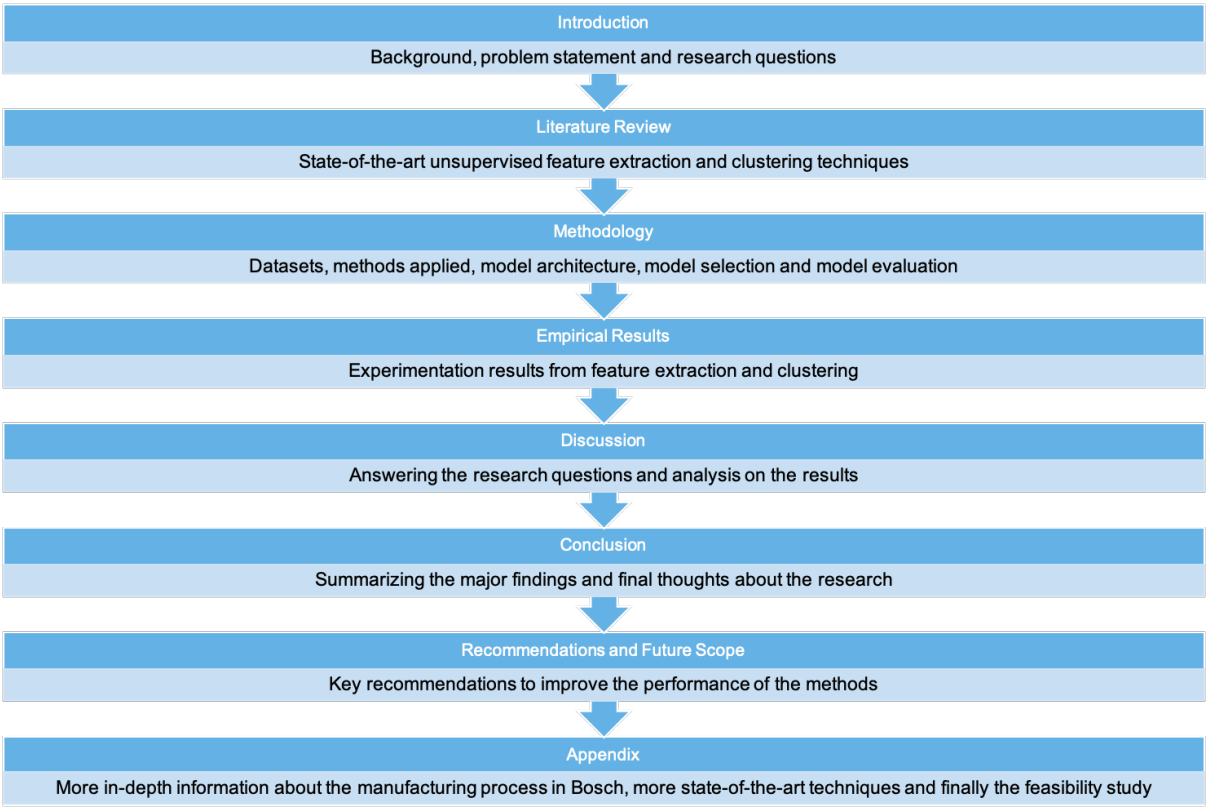


Figure 1.9: A flow diagram summarizing the contents of the thesis

Chapter 2

Literature Survey

2.1 Motivation

As discussed in the previous chapter, the problem faced by Bosch is in tracing a link of unknown origin back to its source. Hence, the type of dataset available is a large set of unlabelled images. The task of this research is to uncover hidden patterns in data without the help of ground truth labels and establish the labels for these links.

Based on the problem statement, a few research questions had been derived in the previous chapter. To address these research questions, it is necessary to conduct a detailed literature survey to identify methods that could either be adapted directly or be customized to meet the requirements of this research. This chapter discusses the related work in the field of unsupervised feature extraction and clustering. Unsupervised feature extraction techniques help to uncover hidden patterns in data without needing the assistance of ground truth labels. Clustering techniques can then be applied to group similar patterns together. Section 2.2 introduces various unsupervised feature extraction techniques. Section 2.3 then presents the different types of traditional clustering techniques and section 2.4 discusses the state-of-the-art deep clustering technique called the DEC.

2.2 Unsupervised Feature Extraction

Unsupervised learning uses a large set of datapoints with no pre-registered labels attached to them. As a result, feature selection in an unsupervised learning problem is a major challenge. However, there are some techniques that help to uncover important features in data without the assistance of any external labels. These techniques are called as unsupervised feature extraction techniques. Some of the commonly used

techniques will be discussed below.

2.2.1 Principal Component Analysis

Principal Component Analysis (PCA), is a conventional non-neural network based feature extraction and dimensionality reduction technique. Using PCA, data in higher dimensional space can be reduced to a lower dimensions.

The dimensions can be reduced in two ways, either by hard-coding the number of components directly or by specifying the percentage of variance to be preserved in data. The reduced dimensions of PCA are called as Principal components. The data is compressed such that the initial few principal components captures the maximum variance in data.

PCA is well suited for unsupervised feature extraction because the technique doesn't need the assistance of external labels to learn features. Given the input data without labels in a higher-dimensional space, PCA constructs the co-variance matrix to identify the relationships among the input feature vectors. From the covariance matrix, an eigendecomposition is performed to identify the eigenvectors and eigenvalues, which give the direction and magnitude respectively of the input feature vectors. With the help of eigenvalues, the composition of the reduced feature space can be decided. Since this whole process happens in an unsupervised manner, the technique is suitable to extract features from a dataset with no labels.

2.2.2 Auto-Encoder

Auto-encoders are a widely used deep learning based technique for unsupervised feature extraction; see [19] for a brief overview. Unlike PCA, they are a non-linear dimensionality reduction technique. An auto-encoder consists of an encoder block where the input data gets compressed to learn latent space representations. These representations are then fed to a decoder block to reconstruct the original data. Both the encoder and decoder are neural networks having a mirror architecture and non-linear activation functions. Figure 2.1 shows a typical block diagram of an auto-encoder.

The learning objective of the auto-encoder is based on the decoder's reconstruction and doesn't need the ground truth labels to learn the feature vectors. Hence, this method is suitable for unsupervised feature extraction.

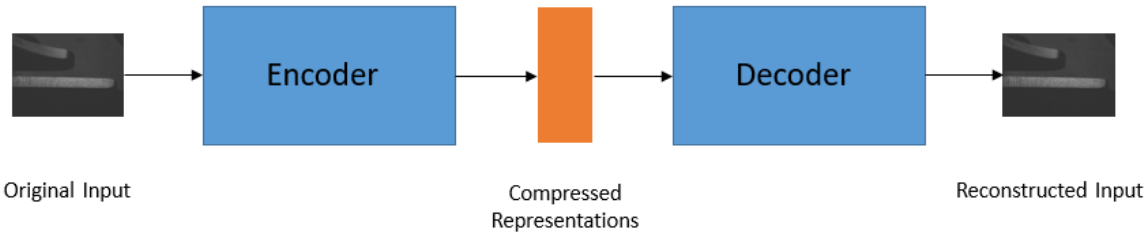


Figure 2.1: The Auto-Encoder model with encoder and decoder as its two components. The output of the decoder is the reconstruction of the input

2.2.3 Variational Auto-Encoder

As discussed earlier, conventional auto-encoders are only capable of learning latent representations and regenerating the same input. However, Variational auto-encoders (VAE; see [15] for a brief overview) are deep generative models that follow a similar architecture as the auto-encoders except for a modification at the bottleneck. The bottleneck now is a distribution in the latent space instead of the representations themselves. These distributions equip VAEs with an additional capability of generating new samples from a given input rather than just input reconstruction.

An auto-encoder that was discussed in the previous section, optimizes a loss function such as the MSE as the reconstruction loss. MSE is the mean squared distance between the reconstructed image and the original image. Here, the latent vector space is discontinuous, that is, if there are clusters present in the feature space, the points will be grouped separately according to the clusters. However, in VAEs, the latent vector space is restricted to be continuous. The latent vectors can then be formed by sampling from this continuous distribution. This property of restricting the latent space makes the VAE capable of generating new samples (a sample that has never been seen before) from the learned representations. A loss function called the Kullback-Liebler Divergence (KLD) which measures the difference between two probabilistic distributions, is optimized so that the learnt representations don't deviate from the latent distribution. In this case, the two distributions are the target distribution and the learnt representational space. An architecture of VAE is shown in Figure 2.2.

The learning objective of a VAE is the same as that of an auto-encoder, with the only difference being how the feature vectors are derived in the latent space, as explained before in this section. Hence, this technique is also suitable for unsupervised feature extraction. However, in VAEs, the latent vectors are derived from a continuous distribution. Thus, there is a possibility of obtaining clusters that are non-discrete and

not well separated.

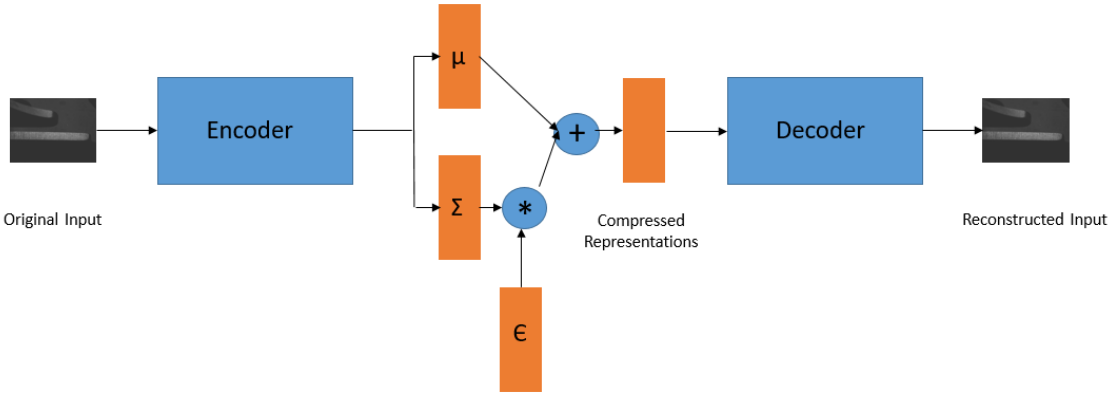


Figure 2.2: Variational Auto-Encoder, a deep generative type of an auto-encoder where the latent vector space is defined by three parameters - μ and $\sum$ are the mean and variance respectively of the latent distribution and ϵ is the randomness parameter

2.2.4 Generative Adversarial Network

Another popular type of deep generative model is Generative Adversarial Networks (GAN; see [9] for a brief overview). GANs comprises of two DNN networks called the generator and discriminator. The generator's task is to generate new samples that are not part of the input. The discriminator's task is to classify images as real or fake. The two blocks play a zero-sum minimax game where the generator's task is to fool the discriminator into accepting a fake sample as real. Both the generator and discriminator learn the features from an image in an unsupervised manner. The discriminator learns the features so that it can discriminate a new sample from the real ones. A generator learns the features to generate new unseen examples. A typical GAN architecture can be seen in Figure 2.3.

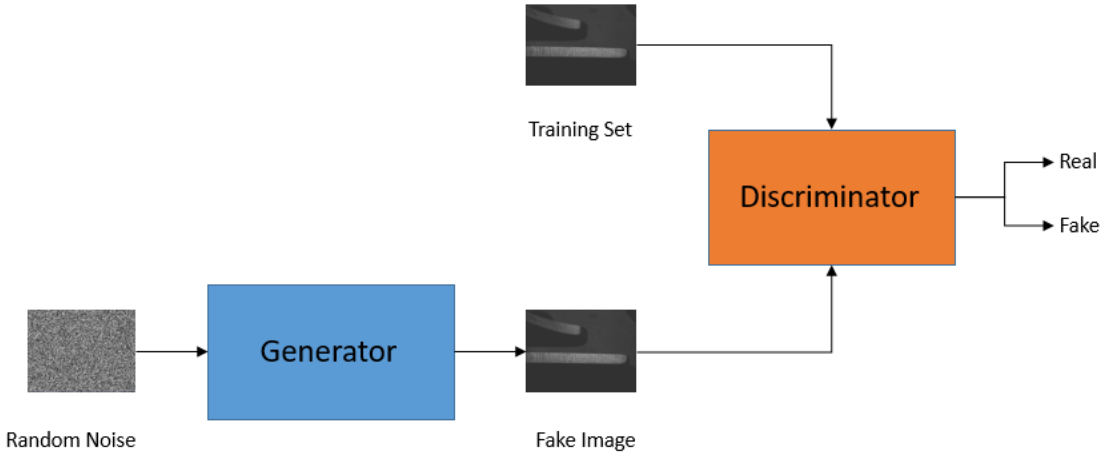


Figure 2.3: Generative Adversarial Networks, a deep generative model comprising of two components, generator and discriminator.

2.3 Clustering Techniques

Clustering is an unsupervised technique to partition data into multiple sub-groups. It is a technique where information learnt from the data is used to group them into clusters based on a similarity metric. Data points within a cluster will be more similar in comparison to data points in different clusters. In this section, some of the widely used traditional clustering techniques will be discussed. A more detailed survey of the different clustering techniques can be found in this paper [1].

2.3.1 K-Means

K-means clustering is a partitional based clustering technique. The algorithm tries to group data points, such that the points belonging to the same cluster are similar. Also, the distance between the points and neighbouring clusters is maximized. The clusters itself are formed based on a set of cluster centroids that are first initialized randomly. The algorithm tries to obtain clusters by minimizing the mean Euclidean distance between the data points and the cluster centroids. Different random initializations might lead to different solutions. One significant drawback of k-means is that it already needs to know the number of clusters in prior. In unsupervised learning, when the number of classes is not known, determining the number of clusters could prove to be a challenge

2.3.2 Gaussian Mixture Models (GMM)

GMM is very similar to k-means with the difference being that GMM assumes data to lie in a mixture of Gaussian distributions. All data points from an independent Gaussian distribution belong to the same cluster. Unlike k-means, GMM updates its clusters using both the mean and standard deviation between the centroids and points in the cluster. Hence, the shape of the clusters doesn't strictly have to be circular. GMM also does a soft assignment of clusters based on the likelihood scores of data points belonging to each cluster. Similar to K-means, GMM also needs the number of clusters to be known in prior.

2.3.3 Agglomerative Clustering

Agglomerative clustering is a type of hierarchical clustering where data points are merged in every iteration based on a hierarchy. That is, all data points are first assigned to their own individual clusters, then these clusters are merged iteratively based on their proximity to obtain a final set of clusters. The major drawback of this technique is that it needs high computation power to perform the iterative merge operation.

2.3.4 DBSCAN

DBSCAN is a density-based clustering algorithm. The algorithm first tries to select an arbitrary starting point and works towards grouping points in the neighbourhood based on the density. Using a distance threshold, points in the neighbourhood are either marked as part of a cluster or just noise. This process is then repeated until all points belong to one of the clusters. The major drawback of this technique is the choice of the distance threshold. There is no trivial way of choosing this value. A large value could result in noisy clusters, while a small value could result in incorrect ones. Unlike K-means and GMM, here there is no need to specify the number of clusters in prior.

2.4 Deep Clustering Techniques

The feature selection techniques discussed before, may not be useful to obtain good clustering performances as their training objective was never to learn clustering-friendly representations. Hence, along with the traditional clustering techniques discussed in the previous section, there is a deep learning based clustering technique whose main training objective is to extract clustering-friendly representations. This section explores the technique called as Deep clustering. [20] [2].

Deep clustering techniques can be widely divided into the following three categories based on their underlying architecture.

1. Auto-Encoder based
2. Variational Auto-Encoder based
3. Generative Adversarial Network based

As the objective of the research is just to learn hidden representations in data, the focus is placed on auto-encoder based techniques rather than deep generative models like VAEs and GANs. Hence, a widely used auto-encoder based technique called Deep Embedded Clustering (DEC) [32] is discussed in the next section. There are also other alternatives for DEC, which are all summarized in detail in Appendix B.

2.4.1 Deep Embedded Clustering (DEC)

Since the choice of the feature space is application dependent, [32] highlights that this choice is to be made by the end-user in most cases. Different perceptions of data might lead to varied results in features. Hence the paper proposes a data-driven approach to solve the feature space and cluster memberships jointly. This method is called as DEC and is said to be significantly less sensitive to the choice of hyper-parameters compared to other state-of-the-art methods. For an unsupervised clustering problem, since the cross-validation is not possible with unlabelled data, this property is a desired one.

As shown in Figure 2.4, DEC has a two-step procedure. Step 1 is to pre-train an auto-encoder to learn an initial set of latent features. These latent features are applied to one of the centroids based traditional clustering algorithms to derive the cluster centroids. After this, the decoder part of the auto-encoder is cut-off. A custom clustering layer is then attached to the output of the encoder, as shown in Figure 2.4. The centroids learnt in the previous step will be initialized as the auxiliary target distribution in the clustering layer.

In step 2, the network of encoder and clustering layer is trained to refine the clusters. At the clustering layer, a student t-distribution kernel is used to calculate the soft assignments called 'q'. 'q' is the likelihood assignment score for each data point in the feature space to all the cluster centroids. Centroids having a large score (scores tending to 1) are said to be high-confidence assignments. The clusters are then iteratively refined by learning from these high-confidence assignments using the auxiliary target

distribution. That is, the KLD loss is optimized between the auxiliary target distribution and the soft assignments 'q'. The auxiliary target distribution is also updated after a certain number of iterations. DEC continuously minimizes the KLD loss until a convergence criterion is met. The criteria is set to a tolerance value of 0.0001 between the current and previous soft assignments. Once this criteria is met, the training stops and the latest predictions are treated as the final cluster assignments.

The paper discusses an evaluation conducted on one text and two image datasets and compares their performance against three state-of-the-art competitors. The results show that DEC outperforms its competition, such as k-means over nine different sets of hyper-parameters on all datasets. The paper also mentions that DEC only shows a linear increase in complexity for an increase in the scale of the dataset, which means they can easily scale up to millions of data points.

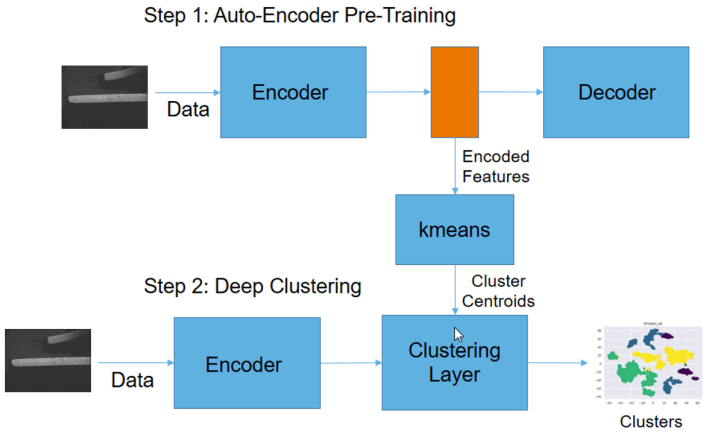


Figure 2.4: An auto-encoder based deep clustering model which does the combined task of feature extraction and clustering using a two-step procedure

After gaining insights from literature, further research will focus on implementing the unsupervised feature extraction, clustering and deep clustering techniques that were introduced in this section.

Chapter 3

Methodology

3.1 Motivation

The previous chapter introduced some of the state-of-the-art techniques that could be useful to solve our research questions. This chapter provides a development pipeline giving a sequence of steps to be performed, starting from the raw data. Out of the four unsupervised feature extraction techniques discussed in the last chapter, this chapter focuses on the two chosen ones for this research, PCA and Auto-encoder. These two are conventional feature extraction techniques whose purpose is to compress data into smaller dimensions. VAE and GAN on the other side are deep generative models mainly used in application areas wherever new data needs to be generated from the given input.

At first, Section 3.2 introduces the development pipeline that shows a transition from raw data to the cluster predictions and evaluation. The next few sections will discuss each step of the pipeline in detail, including an overview of the different data sets. The chapter then concludes with a discussion on the model architecture, model selection and model evaluation.

3.2 Development Pipeline

The development in Figure 3.1, shows a transition from raw data to the clusters. The raw data is of the size 760*1024 which translates into 778240 dimensions. Data in such high dimensions usually contain a lot of background noise. Hence, an elaborate pre-processing step is applied to remove the background noise and bring in proper structure in data. After pre-processing, the size of the images is reduced to 128*1024 that retains only the saddle surface of the link. Though the dimensions are now re-

duced to 131072, they are still significantly high and contains a lot of redundant information.

Various unsupervised feature extraction techniques are then applied to the preprocessed images to reduce the dimensions further and retain only meaningful information such as patterns. The final set of features are then used to train some clustering algorithms. The quality of clusters is measured and compared across various models. The final step is to perform a model validation using the validation dataset with labels collected only for this purpose. Various metrics are used to measure the performance of the cluster space. The performances across different models and clustering algorithms are compared and analyzed. In the next few sections, each step in the pipeline will be discussed in detail.

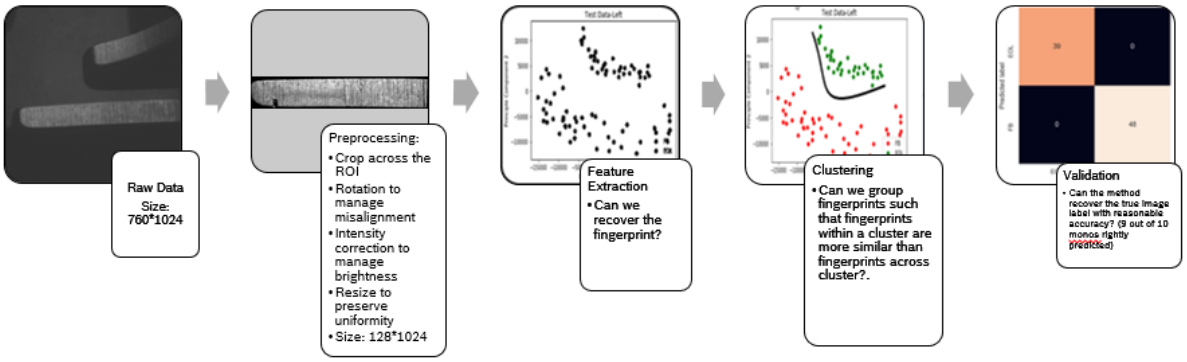


Figure 3.1: The development pipeline that shows a transition from raw data to the clustering results and validation.

3.3 Raw Data

For this research, there are two datasets used, one each for training and validation. Both the datasets have been collected from the same mix. Hence, both have the same mix ID. The first dataset is called the Mix dataset, which is used to train the unsupervised feature extraction and clustering models. The second dataset is called the Mix monsters, which is the validation dataset used to evaluate the models trained using the Mix dataset. Further, both the datasets will be discussed in detail.

3.3.1 Mix Dataset (Training set)

For training purpose, images are collected from one of the mixes with a mix ID of **2043-027**. This mix consists of 9 monos in total. As discussed earlier, each mono contains

approximately 21000 links in them, which results in about 200,000 links for a mix.

Due to several challenges involved in collecting and storing such a large dataset, as discussed in section 1.6, only a small proportion of 5160 images could be randomly sampled from a whole mix. The camera at Station 6 of AOI is configured to capture and store images of the saddle surface area, as shown in Figure 3.2.

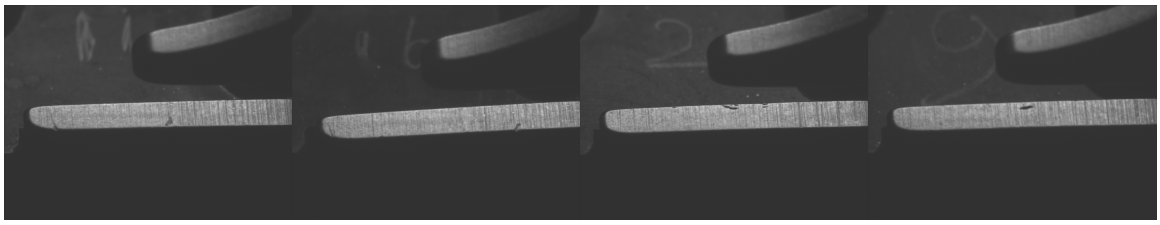


Figure 3.2: Four randomly selected images of the links showing the saddle surface of the links from station 6 of the AOI

3.3.2 Mix Monsters Dataset (Validation set

The second dataset used is called the Mix monsters. They are a small set of labelled images from the same mix as that of the training set images. Hence, they too have the same mix ID of **2043-027**. The mix monsters are collected just before the monos get sent for mixing. A small set of 3 links are randomly sampled from each of the 9 monos, which results in 27 links in total. The images are then clicked for each of these 27 samples by placing them manually at Station 6 of the AOI.

Since the mix monsters are sampled before mixing, all the labels such as the mono ID get preserved. Hence, this dataset can be used for validation of the results obtained from the training set. That is, these mix monsters can be used to perform external cluster evaluation on the clustering models that are trained using the mix dataset.

3.4 Pre-processing

Pre-processing is an important step that intends to eliminate any background noise that may hamper the algorithm's performance and speed. The first step is to crop off the background by retaining only the region of interest form the raw image. The ROI is the region where the grooves on the saddle surface lie as shown in Figure 3.3. An initial cropping is first done by applying a bounding box estimated to fit the ROI inside the box. Then an edge detection technique called the Canny edge detector is used to further adjust the bounding box to perfectly fit the saddle surface and crop again

to this adjusted bounding box. Rotation is performed on all images to normalize the alignment and are also intensity corrected to reduce the influence of lighting. Finally, the images are normalized by dividing each pixel value by 255.

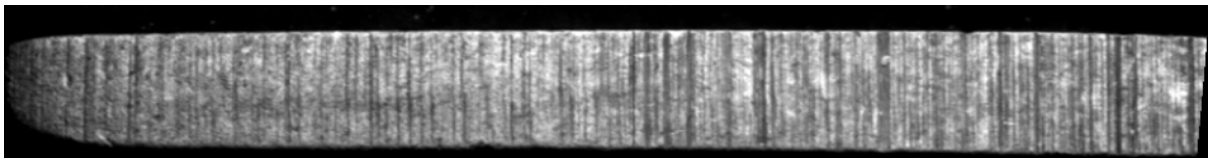


Figure 3.3: The resulting Region of Interest (ROI) after applying the elaborate pre-processing steps on the raw image obtained from station 6 of the AOI

3.5 Model Architecture

The underlying architectures of the two feature extraction techniques, PCA and auto-encoders will be discussed in this section. This section explains how the input data is converted to a compressed feature space using those two methods.

3.5.1 PCA

The PCA, as discussed earlier, is a dimensionality reduction technique that tries compress data in a higher-dimensional space to a lower-dimensional one by preserving the maximum variance in them. The idea is, to first apply this simple technique to establish a bench-mark in terms of the results. These bench-marked results would later help to estimate how beneficial was it to move towards a complex neural network-based algorithm like auto-encoder.

The pre-processed images are of size 128*1024, which translates into a dimensional space of 131,072. After applying PCA, the dimensions get reduced depending on the number of principal components specified.

3.5.2 Auto-Encoders

PCA is a linear model and relies on an assumption that data always lies in a linear space. This assumption might not always hold true, especially with images of larger dimensions where the intricate details usually lie in a non-linear space. Hence, it is often good to look for non-linear alternatives such as neural networks to learn representations. Neural-networks are built using multiple neurons across multiple layers. Each neuron is capable of computing a complex non-linear activation function. This

capability could prove useful to learn hidden patterns in images. One of the major neural network based unsupervised feature extraction technique is an Auto-encoder.

As previously discussed, auto-encoders also have reconstruction capability. So, it is easy to visualize how well the learnt features can be used to reconstruct the original image. Thus the training objective is to first optimize the mean-squared error (MSE) between the reconstructed image and the original image. Lesser the MSE value, better the reconstruction. After training the model based on the MSE, the ultimate objective is to apply these features on clustering algorithms and optimize their clustering performance.

In the network, multiple sets of convolutional and pooling layers are used in combination to reduce the very-large dimensions of input images to a small number. The combination also helps to learn the local structures in data very effectively. At the bottleneck, a set of dense layers are added to integrate the local level features into one set of global features.

3.6 Model Selection

To choose the right model with the right values for the hyper-parameters, an elaborate model selection process was conducted. As the objective of the research is to find similar patterns in data, the best model is the one which gives the best clustering performance. The two types of clustering performance metrics will be discussed later in this chapter.

There are different sets of parameters for both PCA and auto-encoders. The next sections provide a detailed model selection procedure for both PCA and auto-encoders.

3.6.1 PCA

There are two tunable parameters for PCA as discussed below.

- **Variance**: This parameter specifies the percentage of variance to capture in the resulting principal components.
- **Number of principal components**: Another way to reduce the dimensions is to already specify the number of principal components desired.

To choose the right number of principal components, a scree plot of the cumulative explained variance versus the number of components can be obtained. Figure 3.4 shows the scree plot for Mix dataset. From the plot, it can be observed that the initial set of principal components captures most of the variance.

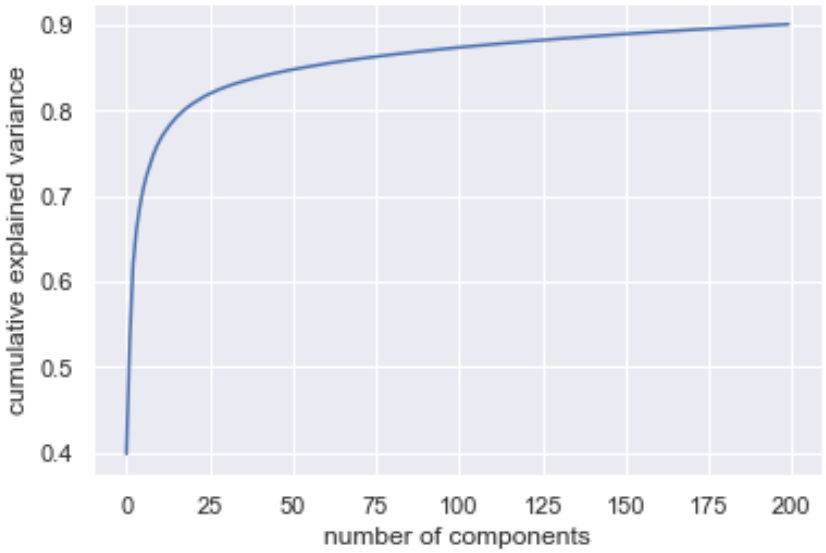


Figure 3.4: A Scree plot for the Mix dataset: A plot of number of features against the cumulative explained variance obtained from PCA. The initial few components preserve the most variance in data.

Based on the scree plot, the following set of components are experimented as shown in Table 3.1, to select the one that gives the best clustering result. 5159 was the maximum number of features obtained with 99% of variance, which is also equal to number of samples-1.

Model ID	# of Features
1	3
2	5
3	10
4	100
5	200
6	5159

Table 3.1: The list of feature sizes to be experimented on PCA and auto-encoder

3.6.2 Auto-Encoders

Auto-encoders have several hyper-parameters and values that can be tuned, some of which are discussed below. It is impossible to exhaustively train all the possible options for each hyper-parameter. Also, it is very time-consuming and computationally intensive to do a combination tuning of all the parameters. Hence, the approach chosen was to tune each hyper-parameter individually by fixing the other ones to their default values. It was noticed that the default hyper-parameters provided the best clustering performance. However, the epochs, batch size, number of encoded features and the channel sizes don't have a default value. Thus, they still needs to be tuned using different values.

- **Channel sizes**: This parameter decides the number of channels to be applied at each convolutional layer. Different combinations were tried but the best results were observed for the channel sizes of [4, 8, 16, 32, 64] from the input layer to the bottleneck.
- **Number of encoded features**: This parameter defines the number of features at the output of the encoder or the bottle-neck of the auto-encoder. This is particularly an important parameter as these encoded features will then be later used to perform the clustering task. Hence, the clustering performance is completely dependent on this value. As the scree plot cannot be plotted for an auto-encoder, various values same as the PCA as shown in Table 3.1 were experimented.
- **Optimizer**: This parameter influences the speed and performance of the training procedure. Some of the options including Adam, Momentum, Adagrad and RMSprop were tried. Through the individual tuning procedure, it was observed that Adam [5] provided the best clustering performance.
- **Loss function**: This parameter defines the loss function to be optimized. In the case of an auto-encoder, this function is the mean-squared error (MSE) between the reconstructed and the original images.
- **Batch size**: This parameter gives the size of the batch to be used for training at each epoch. Again various values were tried and it was observed that 8 provided the best clustering performance.
- **Epochs**: This parameter decides the number of iterations the training needs to be carried out. A value of 20 was set but different numbers ranging from 10 to 200 were also explored. It was found that after 20 the loss function flattens out as seen in Figure.

- **Activation function**: This parameter is kept with one of the most widely used and also very consistent activation function ReLu in all the layers [26]. It was also observed to be the best in the individual tuning procedure.
- **Initializer**: This parameter is kept with the default value of Glorot uniform initializer which was also found to be the best in the individual tuning procedure.

Deep Clustering (DC)

- **Loss function**: This parameter defines the loss function to be optimized for deep clustering. A KLD loss is used to minimize the loss between an auxiliary target distribution and predicted soft assignments of the cluster centroids.
- **Number of clusters**: This parameter is used to create the number of units/neurons in the custom clustering layer.
- **Initializer**: This parameter is used to initialize the cluster centroids for the custom clustering layer. Any clustering algorithms that can learn a set of cluster centroids could be used for this purpose, for example, K-means or GMM.

3.7 Model Training

The feature extraction and the clustering models are first trained using the large Mix data of 5160 unlabelled images. For a PCA model, this training is straight forward and is done by just providing the number of principal components required from the PCA model. For an auto-encoder, however, an elaborate training procedure needs to be performed to learn the model weights. During the training procedure, the MSE loss function is optimized at every epoch and is allowed to do so for 20 epochs. Once the feature weights are trained, the next task is to train the weights of the clustering algorithms K-means and deep clustering. The weights of K-means are initialized using 'k-means++' initializer. The weights of deep clustering are initialized with the centroids learnt from K-means. Since K-means is largely sensitive to centroid initialization, the algorithm is allowed to train with 100 different restarts.

3.8 Model Evaluation

After training the models, the final step is to perform a model evaluation. The two types of evaluations that will be performed are feature and cluster evaluation. Fea-

ture evaluation deals with the measurement of how well the compressed features can reconstruct the original input. The cluster evaluation deals with the measurement of quality and correctness of the resulting cluster predictions.

3.8.1 Feature Evaluation

In a supervised task, the ground truth can be used to evaluate the quality of the resulting feature space. But for an unsupervised task, this validation is not possible as there are no labels. An alternative way to test the quality of the feature space is by reconstructing the original data using the resulting feature space. A feature space which has the capability of reconstructing the original image well is believed to be meaningful. For this purpose, auto-encoders could be used. An auto-encoder tries to optimize the error between the reconstruction and the original data iteratively.

3.8.2 Cluster Evaluation

The quality of the resulting clusters can be evaluated using one of the following two approaches:

- **Internal evaluation**: This is a type of evaluation that can be performed on unlabelled datasets to test how well separated and compact are the resulting clusters. Three methods that can be used for this purpose are Silhouette coefficient, Calinski Harabasz index or Davies Boulding index.
- **External evaluation**: This is a type of evaluation that can be performed with labelled datasets to measure how well the cluster preditions comply with the ground truth labels. Some of the methods that could be used for this purpose would be V-measure, Adjusted Rand index (ARI), Fowlkes-Mallows Index (FMI) scores or mutual information score based approaches.

The resulting clusters are first tested for quality using internal evaluation, using the large unlabeled mix dataset. Then, an external evaluation was performed using the small-sized mix monsters labelled dataset. Both these measures together help to determine how well constructed and correct are the obtained clusters.

3.9 Performance Metrics

For the different types of model evaluation methods discussed in the previous section, various performance metrics for each method, that will be used to evaluate the results will be discussed below.

3.9.1 Feature evaluation metric

Mean-Squared Error (MSE)

The Mean-Squared Error is calculated using the original values of the image and its reconstructed values from the auto-encoder. Hence the MSE is given by the mean of the differences in pixel-wise intensities between the original image and the reconstructed image over the entire dataset.

$$\frac{1}{n}\Sigma_{i=1}^{n}(y_i true - y_i pred)^2 \tag{3.1}$$

- n: The total number of datapoints.
- ytrue: The intensity values of the original image.
- ypred: The intensity values of the reconstructed image.

The error is minimum when the values tend to 0.

Unlike other metrics like the mean-absolute error, the MSE calculates the squared error between the pixel intensities. Hence, large differences in the pixel intensities between the output and the input are severely penalized. Also, as the images in the mix dataset have been normalized to lie in a range between 0 and 1, the absolute error for such a range will always be very small. But, in order to provide a better feedback mechanism to the auto-encoder model, it is better to use a squared error which magnifies an error value. However, after squaring the pixel values, the MSE error will not retain the unit of the pixel intensities.

3.9.2 Internal evaluation metrics

Internal evaluation methods will be used to evaluate the performance of clusters obtained from the unlabelled Mix training set.

Silhouette coefficient

The Silhouette coefficient is calculated using the following two variables

- a: The mean distance between a sample and all other points in the same cluster.
- b: The mean distance between a sample and all other points in the next nearest cluster.

Hence for each sample, the coefficient is defined by:

$$S = \frac{b - a}{max(b, a)} \tag{3.2}$$

Finally, the Silhouette Coefficient for an entire set is calculated by taking the mean of the Silhouette coefficients for each sample. The values lie between -1 and +1. Values tending to 1 indicate good clustering while negative ones indicate bad clusters or wrong cluster assignments. Values that tend to 0 are seen to have overlapping clusters.

The advantage of Silhouette scores is that it is also based on the distance metric, similar to the clustering algorithms used. Hence, it is a true measure of clustering quality. Also, unlike other internal clustering quality metrics such as the Davies-Bouldin index, with the Silhouette coefficient, it is possible to measure the quality in terms of these three possible results, whether the clusters are well separated, overlapping or incorrect.

However, the major drawback is it is computationally expensive to compute this score. And also similar to some of the other internal cluster evaluation metrics, the metric performs better with convex clusters obtained from methods like K-means and hence are not suited for density-based clusters.

3.9.3 External evaluation metrics

External evaluation methods are used to evaluate the performance of labelled datasets such as the feasibility dataset and mix monsters.

Homogeneity, completeness and V-measure

- Homogeneity: Clustering results have good homogeneity scores when all of the clusters have data points from only one true label.
- Completeness: Clustering results have good completeness score when all points from a single true label belong to the same cluster.
- V-measure is the harmonic mean of homogeneity and completeness.

$$Vmeasure = 2\frac{hc}{h + c} \tag{3.3}$$

  - h: Homogeneity score of the clustering result

- c: Completeness score of the clustering result

A value of 1 indicates that the predictions match the ground truth labels perfectly.

The main advantage of this metric is that it is independent of the clustering algorithm, dataset, number of classes or clusters used to obtain the cluster predictions. And it is a much better evaluation metric compared to its alternatives because of its ability to measure a combination of two scores harmonically. The two scores, homogeneity and completeness are both the most desirable factors in clustering solutions. The metric can also be used as a diagnostic tool to gain better insights into the clustering results based on two scores.

Similar to other external cluster evaluation metrics such as the Fowlkes-Mallows index, the main disadvantage of this metric is that a cluster to class mapping cannot be derived from these metrics. Hence, they are only useful to measure the quality of cluster predictions.

Chapter 4

Empirical Results

4.1 Motivation

Having already discussed the three different methods PCA, AE and Deep clustering for unsupervised learning, this chapter aims to provide the results obtained for each method. The results from feature evaluation, internal cluster evaluation and external cluster evaluation will be presented for different models against different clustering algorithms.

4.2 Feature Evaluation

The first results shown is the feature evaluation of the auto-encoder model experimented with different sizes of the feature space.

4.2.1 Mean-Squared Error (MSE)

As discussed earlier, MSE is a useful loss function to train an auto-encoder to optimize the reconstruction. Different models of an auto-encoder were experimented with different number of encoded features, as listed in the Table 4.1.

A graph of number of epochs versus MSE for one of the models is shown in Figure 4.1. The graph indicates that there is no drastic improvement in the MSE after 20 epochs. Hence, the MSE values for each of these models are captured at the end of 20 epochs. It can be observed that as the number of features is increased, the MSE score also improves. This indicates that better reconstruction of the input image can be obtained with more number of features.

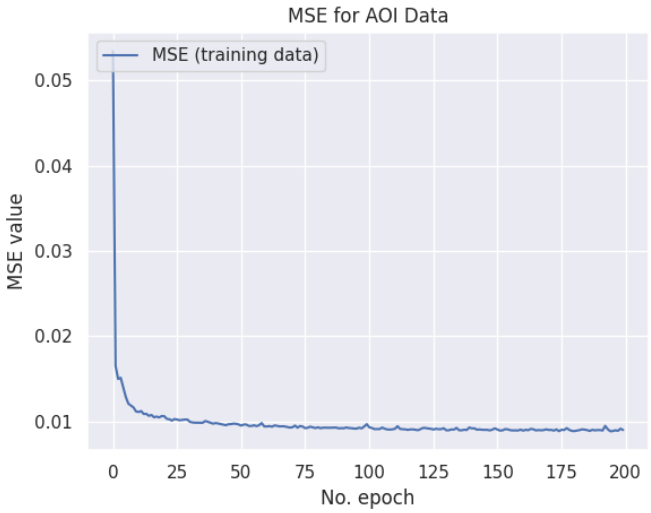


Figure 4.1: A plot of number of epochs vs MSE for the auto-encoder model trained with the mix dataset

Method	# of Encoded Features	MSE
1	3	0.0129
2	5	0.0107
3	10	0.0102
4	100	0.0097
5	200	0.0097
6	5159	0.0027

Table 4.1: The MSE scores for the auto-encoder model with different number of encoded features

A graphical representation of Table 4.1 can be seen in Figure 4.2. The figure shows a plot of number of features against the MSE scores obtained for each.

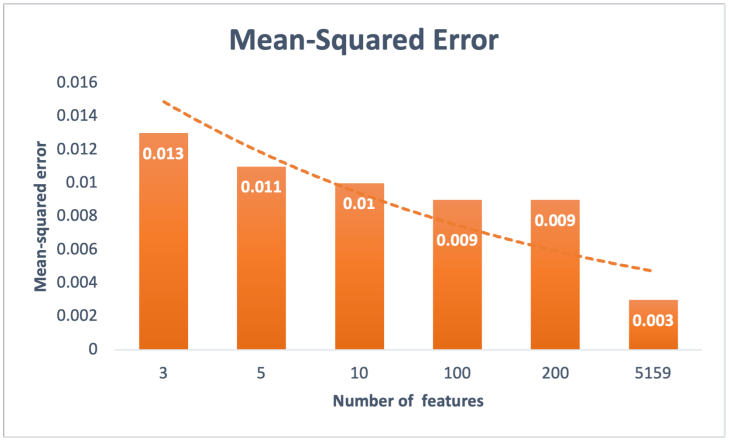


Figure 4.2: A graphical representation of the number of features against the MSE scores for the auto-encoder model

4.3 Cluster Evaluation

The resulting compressed features space from PCA and auto-encoder from the training set are used to train the clustering models such as K-means. The trained clustering models are then used to predict the cluster assignments for the test data. At last, a cluster evaluation is performed to measure the quality and correctness of the resulting clusters. The internal and external cluster evaluation metrics are used respectively for this purpose.

Since the training data has no label information attached to them, it is only possible to perform an internal evaluation on their cluster predictions. The Silhouette coefficient is useful for this purpose. The quality of clusters in terms of how tightly packed and well-separate are the clusters can be measured using this technique.

An external cluster evaluation was also conducted using the labelled validation dataset. The external scores help to measure the correctness of the obtained clusters or in other words, they measure how closely the predicted labels match the true labels.

4.3.1 Principal Component Analysis (PCA)

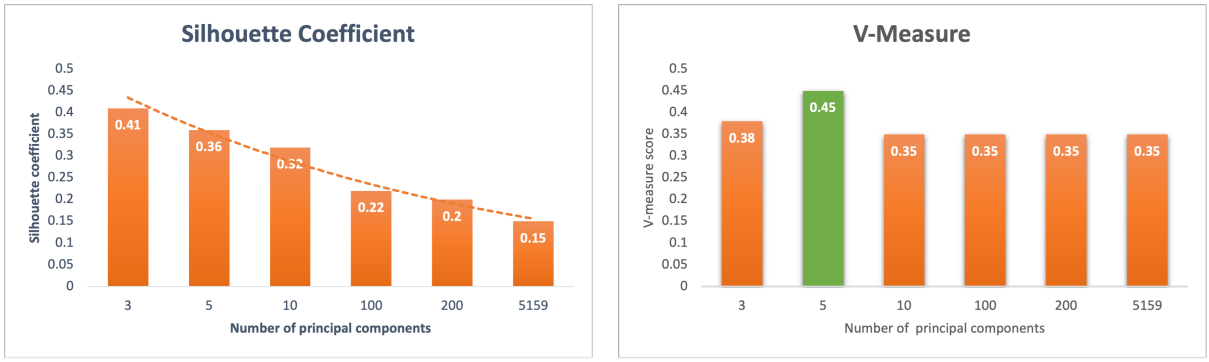
The Silhouette scores obtained after applying K-means for clustering the PCA components are shown in Table 4.2. It can be observed that as the number of dimensions increase, the Silhouette score reduces.

Different models were validated based on the V-measure scores to choose the model with the best score. The scores obtained after applying K-means on the different number of PCA components are shown in Table 4.2. The table indicates that the score returns a peak of 0.45 with 5 principal components. From 10 components and further, an increase in the number of components didn't affect the V-measure.

A graphical representation of the scores in Table 4.2 is shown in Figure 4.3. The Figure shows plots of the Silhouette and V-measure scores obtained for different number of features.

Model	# of Principal Components	Silhouette Coefficient	V-Measure
1	3	0.41	0.38
2	5	0.36	0.45
3	10	0.32	0.35
4	100	0.22	0.35
5	200	0.20	0.35
6	5159	0.15	0.35

Table 4.2: Cluster evaluation scores for the PCA+K-means model with different number of principal components



(a) Silhouette Coefficient

(b) V-Measure

Figure 4.3: Graphical representations of number of features vs cluster evaluation scores for the PCA+K-means model (a) A plot of the Silhouette Coefficient scores (b) A plot of the V-measure scores

4.3.2 Auto-Encoders

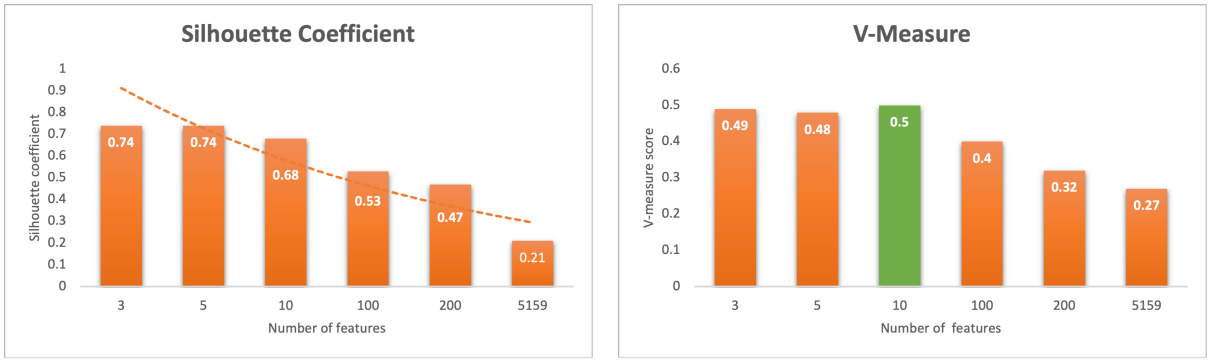
The Silhouette scores from auto-encoder and K-means are also shown in Table 4.3. A similar trend of the Silhouette scores declining with the increase in the number of features, as seen in PCA, was also observed here.

The V-measure scores for auto-encoder with k-means models are shown in Table 4.3. It can be observed from the table that the model with 10 encoded features produced the best V-measure of 0.50. Further increase in the number of features produced lower scores.

A graphical representation of the scores in Table 4.3 is shown in Figure 4.4. The Figure shows plots of the Silhouette and V-measure scores obtained for different number of features.

Model	# of Encoded Features	Silhouette Coefficient	V-Measure
1	3	0.74	0.49
2	5	0.74	0.48
3	10	0.68	0.50
4	100	0.53	0.4
5	200	0.47	0.32
6	5159	0.21	0.27

Table 4.3: Cluster evaluation scores for the Auto-encoder+K-means model with different number of encoded features



(a) Silhouette Coefficient

(b) V-Measure

Figure 4.4: Graphical representations of number of features vs cluster evaluation scores for the Auto-encoder+K-means model (a) A plot of the Silhouette Coefficient scores (b) A plot of the V-measure scores

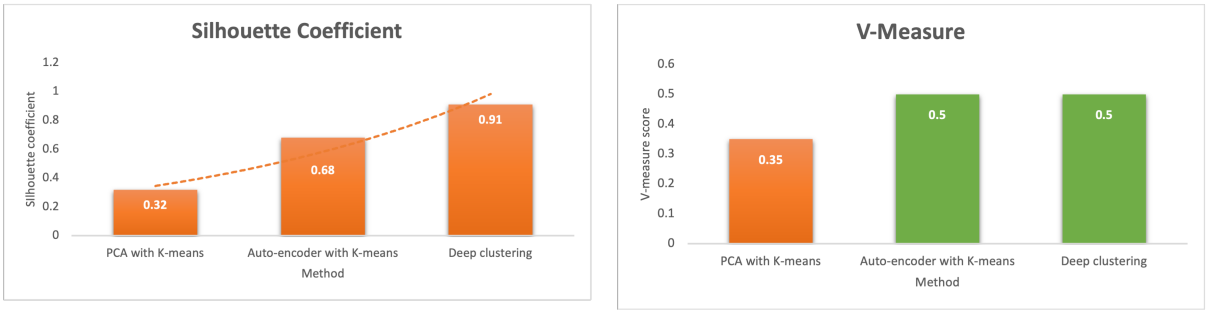
4.4 Deep Clustering

The auto-encoder model with 10 features, was also experimented with Deep clustering, a state-of-the-art clustering method based on deep learning. The clustering quality for this model was observed to improve significantly, resulting in a Silhouette score of 0.91. This was found to be the best score observed across all the algorithms discussed.

Model	# of Encoded Features	Silhouette Coefficient	V-measure
3	10	0.91	0.50

Table 4.4: Cluster evaluation scores for the deep clustering model with 10 encoded features

4.5 Comparison across different feature extraction algorithms

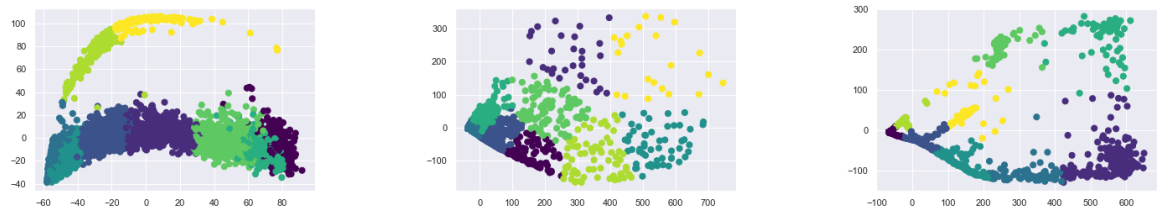


(a) Silhouette Coefficient

(b) V-Measure

Figure 4.5: A comparison of cluster evaluation scores for the different feature extraction methods (a) A plot of the Silhouette Coefficient scores (b) A plot of the V-measure scores

In Figure 4.6, the clusters obtained from different feature extraction methods are visualized in 2D. Specifically, Figures 4.6a and 4.6b show the clustering result of applying 10 features from PCA and auto-encoder respectively on the K-means algorithm. The clustering result of Deep clustering, which is a hybrid method that performs both feature extraction and clustering is shown in Figure 4.6c.



(a) PCA Clusters

(b) Auto-encoder Clusters

(c) Deep Clustering Clusters

Figure 4.6: Cluster visualization for different feature extraction methods (a) Cluster results with PCA features (b) Cluster results with auto-encoder features (c) Cluster results with deep clustering features

4.6 Comparison across different clustering algorithms

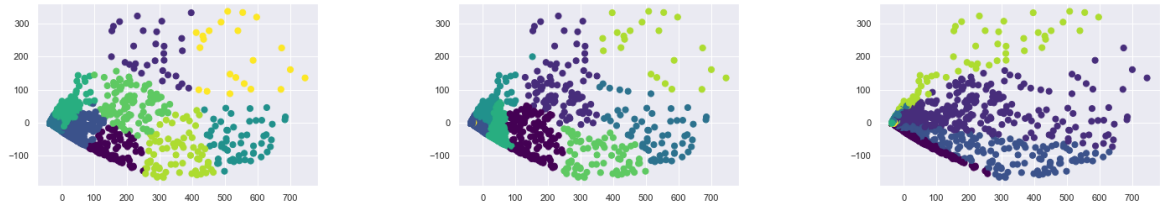
With K-means, an auto-encoder model with 10 features resulted in the best V-measure score in comparison to the other models as seen in Table 4.4. The Silhouette score was observed to be 0.68 for this model. In this section, the results from some of the other clustering algorithms that were experimented on the same auto-encoder model will be shown.

Since the number of clusters is already known based on domain knowledge, only the clustering algorithms where the number of clusters can be explicitly specified were chosen. The algorithms experimented were Agglomerative clustering, a hierarchical based clustering method and GMM clustering. It was observed that Agglomerative and K-means clustering resulted in identical clustering quality. However, GMM performed very badly with a score of just -0.17. A negative Silhouette score indicates that there are wrong assignments of the cluster labels in the predictions. For instance, a sample that is predicted to belong to one cluster is found to be more closer to its neighbouring cluster. Hence, that point should have been predicted to belong to the neighbouring cluster instead. The scores resulting from different clustering algorithms using features from an auto-encoder model with 10 features are presented in Table 4.5.

Model	# of Encoded Features	Silhouette Coefficient		
		K-means	Agglomerative	GMM
3	10	0.68	0.68	-0.17

Table 4.5: Performance of other clustering algorithms against K-means

Figure 4.7 displays the clustering results of three algorithms K-means, Agglomerative and GMM in 2-D. It is observed that K-means and Agglomerative clustering, as indicated by their Silhouette scores in Table 4.5, have very similar cluster representation in 2D.



(a) K-means Clusters *(b)* Agglomerative Clusters *(c)* GMM Clusters

Figure 4.7: Cluster visualization from different clustering methods (a) Cluster results with K-means (b) Cluster results with Agglomerative (c) Cluster results with GMM

4.7 Clustering Mapping

The cluster labels predicted by the clustering algorithms are randomly generated. Therefore, it is not accurate to directly compare the cluster predictions to the true labels. The cluster labels need to be mapped to class labels by first deriving all the possible permutations for the cluster predictions. Each of the permutations is then

compared to the true labels to measure the accuracy score. The mapping with the highest accuracy score is the final mapped predictions.

The true labels and the predicted cluster labels from the deep clustering model discussed in section 4.4 are as follows-

True labels:
[0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7, 8, 8, 8]
Original cluster labels (Before mapping):
[7, 2, 4, 4, 4, 2, 1, 1, 3, 2, 0, 2, 4, 4, 4, 1, 4, 5, 1, 2, 2, 1, 7, 5, 1, 3, 7]
New cluster labels (After mapping):
[7, 6, 4, 4, 4, 6, 2, 2, 8, 6, 3, 6, 4, 4, 4, 2, 4, 5, 2, 6, 6, 2, 7, 5, 2, 8, 7]

The confusion matrices for the above values are shown in the Figure 4.8. The figure includes the confusion matrix before mapping (with the original cluster labels), a table showing the mapping from the original cluster labels to the mapped labels and the confusion matrix after mapping (with the new mapped labels). The accuracy was observed to improve from 0.19 before mapping to 0.41 after mapping.

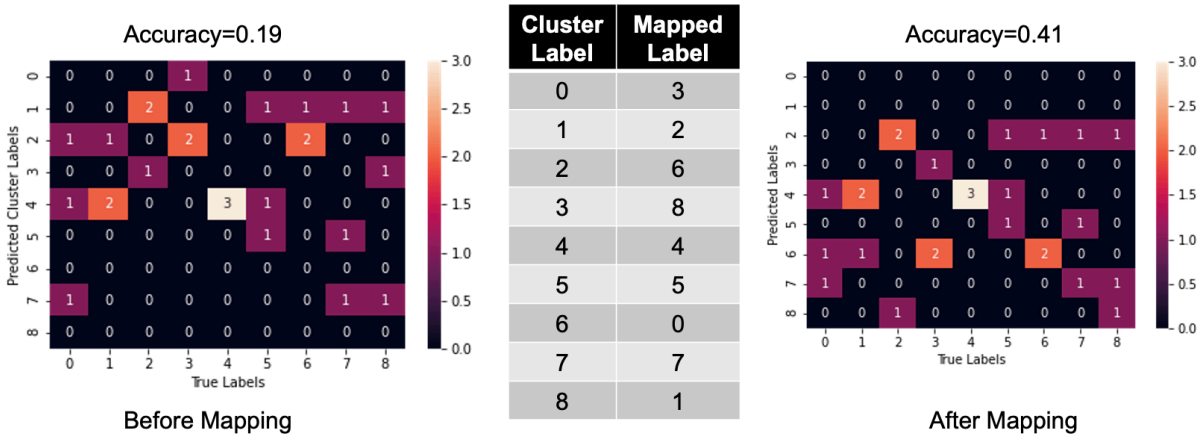


Figure 4.8: A comparison of the confusion matrices before and after mapping of the cluster labels

Chapter 5

Discussion

5.1 Motivation

This section aims to discuss and analyze the results presented in the previous chapter. The chapter starts with a section to answer the research questions stated in the introduction of this paper. Once the research questions are addressed. Finally, the chapter concludes by listing the challenges and limitations encountered during the research.

5.2 Answering the research questions

Main Research Question: How do we extract the features left by the process parameters on the links, and then trace them back to their source?
Ans: The task of this research was to identify ways to extract information that could help to trace the links back to their source. Various techniques were used to extract the information from the images. The techniques were able to extract the information in an automated manner, without the requirement of manual feature engineering. Then clustering techniques were used to group similar data points together. Internal cluster validation techniques were used to measure the quality of the obtained clusters. It was observed that the quality of clusters was very good especially with a technique called deep clustering, which combines feature extraction and clustering technique. Unlike other methods such as auto-encoders where the extracted features are frozen before applying them to a clustering algorithm, deep clustering iteratively refines the clusters and feature space during its training phase, with an objective to obtain well defined and well-separated clusters.

It was then expected that all samples belonging to the same mono label will also form the same cluster(Completeness) and samples part of the same cluster are all be-

longing to the same mono label(Homogeneity). To prove this theory, a small validation set was used to match the cluster predictions to the ground truth mono labels using an external cluster validation metric. However, the external validation scores proved to be inconclusive to trace the cluster predictions of the samples to their ground truth labels. This is thought to be mainly due to an insufficient and imbalanced dataset. Detailed recommendations are provided later in this report to improve the performance of the discussed methods and results.

RQ1: Given unlabeled images of links, how do we optimally extract the features using Deep Learning?
Ans: The images obtained were in a raw unstructured format. An elaborate pre-processing procedure was conducted to remove the background noise and only retain the most important information in those images. Various unsupervised techniques were then applied to the pre-processed images to reduce their dimensions. A detailed discussion about different unsupervised feature extraction techniques is provided in section 2.2. This step is crucial to overcome the curse of dimensionality and retain only the most important information in data. As the number of dimensions to reduce to is still an open question for unsupervised learning, different possible values were experimented with an aim to identify the right choice of dimensions that gives the best clustering performance.

RQ2: Given the set of learnt features, how to optimally cluster them into meaningful subgroups?
Ans: The features learnt from the unsupervised feature extraction algorithms are then applied to one of the clustering algorithms to obtain sub-groups based on similarity. Different clustering algorithms can be used for this purpose as seen in section 2.3. The number of monos present in this mix was already known to be 9. Through this piece of domain information, the number of clusters was fixed to be 9. Hence, only the clustering algorithms where the number of clusters could be explicitly specified such as K-means, Agglomerative clustering and GMM were chosen. Additionally, a more sophisticated deep learning based clustering model called DEC, was also experimented to learn a refined set of features that produced better quality clusters.

However, the resulting clusters still need to be analyzed to identify the best meaning out of them. A small sample set of labelled data from the same mix was used to match the resulting clusters to the labelled samples.

RQ3: How does the choice of the feature space affect the reconstruction and the

clustering performance?
Ans: From the results, it is clear that an auto-encoder model with good reconstruction does not necessarily imply that the clustering performance will also be good. For instance, from Table 4.1, it can be observed that a model with 5159 features produces the best MSE score. But clustering performance scores for the same model are the worst in comparison to other models, as can be observed in Table 4.3. Having more features would mean that the data compression rate is very less. A large uncompressed feature space would still preserve the finer details of an image. Hence, it is simpler for the decoder to reconstruct the input image better. However, it is seen that the clustering performances improve only when the compression rate is increased.

To determine the right compression rate that leads to optimal clustering performance, various sizes for the feature space have been experimented with. In the next section of this chapter, it will be explained why the internal evaluation (Silhouette Coefficient) scores deteriorate when the size of the feature space is increased. Based on those results, it can be concluded that the Silhouette scores and the MSE are negatively correlated in terms of the number of features used.

RQ4: Given that the clusters are detected, how can they be validated to establish their source?
Ans: From the results discussed in the previous chapter, it is clear that the clustering algorithms do indicate the presence of clusters in the mix data. Especially, a high Silhouette score from deep clustering indicates the presence of also well defined and well-separated clusters. As clusters themselves are formed based on similarity among data points, there is a unique feature causing the points to belong to the same clusters.

Hence, it was encouraging to perform an external evaluation with labelled data to prove the meaning of these clusters. The external evaluation revealed that the mapping of the clusters to a mono label was only moderately correct. The best mapping obtained had a correctness score (V-measure) of just 0.5. Which means, only 50% of the whole cluster predictions could be correctly matched to their true label. Hence, though the existence of clusters was proven in the mix data, it was still inconclusive to establish a link between the clusters and the source (the source, in this case, is the monos).

5.3 Result Interpretations

One of the major challenges of this research was to find the right size of the compressed feature space. Similar challenges have been encountered in the past and have been discussed in the literature, as seen in this paper [25]. The paper highlights the importance of feature selection from high dimensional data for clustering algorithms. It is observed data in high dimensions completely masks the clusters. That is, each point in the dataset is observed to be nearly equidistant from one other in a high dimensional space such that the clusters are no longer visible. These equidistant points thus confuse the clustering algorithms that work based on a distance metric to solve the clustering problem.

Another issue discussed in the paper is the curse of dimensionality on the clustering algorithms. An experiment is conducted to demonstrate the effect of the number of dimensions on the distribution of points. For visualization purpose, only 1, 2 and 3 dimensions are experimented on. It was observed that increasing the number of dimensions caused the points to spread out more. It is also thus theorized that further increase in dimensions will cause the points to become more spread out before reaching a stage where they are all equidistant from each other, with further increase in the dimensions causing no more effect.

The clustering quality scores (Silhouette scores) obtained for the different number of dimensions from two different dimensionality reduction methods are presented in Tables 4.2 and 4.3. It can be observed that the increase in the number of dimensions hampered the Silhouette scores. This can be explained to be caused due to the curse of dimensionality discussed above, where the points become more spread out as the number of dimensions are increased. For the Silhouette scores to be high, the mean intra-cluster distance needs to be low. If the points are spread out, this distance will be high, hence affecting the Silhouette score.

5.4 Additional Experiments

In addition to the main experiments discussed in the main part of this report, there were also a few additional experiments conducted during the initial phase of the research. These additional experiments are described in detail in the Appendix. Initially, a feasibility test was conducted using simple methods like PCA and T-SNE to verify if there was a direct influence of two process parameters (runs and track) on the groove

patterns. However, this experiment proved to be inconclusive mainly due to the nature of the methods used for feature extraction and the small size of the dataset.

Moreover, a 2D and 1D correlation were performed on the labelled datasets to verify if a strong correlation exists among data samples belonging to the same process parameter. But this experiment also failed to conclusively prove that the correlation exists. However, these additional experiments still highlighted the importance of collecting a larger dataset and the need to move towards more complicated deep learning based techniques for feature extraction. A detailed description of both these experiments is provided in Appendix C.

Chapter 6

Conclusion

The primary goal of the research was to prove the existence of unique patterns in the mix data. It was largely successful in proving this hypothesis, as most of the clustering algorithms revealed the existence of clusters through good internal cluster evaluation scores. But further attempt to trace these clusters to the mono IDs through an external evaluation using a small sample-sized validation set proved to be inconclusive.

However, some of the interesting observations were that the clusters from auto-encoder had a better clustering quality than that from PCA. It was also observed that deep clustering, which is combined feature extraction and clustering technique, performed the best against various stand-alone feature extraction and clustering algorithms. The choice of the feature space proved to be the most challenging aspect of the research. For an auto-encoder, it was observed that the internal cluster evaluation scores were negatively correlated to the mean-squared error in terms of the size of the feature space. That is, as the size of the feature space increased, the Silhouette scores deteriorated, while the mean-squared error improved.

Finally, there are definitive hints to show that the mix data contains clusters and thus is not uniformly distributed. However, there needs to be a better design in the data collection in order to improve the validation scores and eventually have a reliable traceability application. The next chapter provides the recommendations to help achieve the same.

Chapter 7

Recommendations and Future Work

7.1 Motivation

Having provided the conclusion and limitations before, this chapter aims to provide the recommendations and discuss the future scope in this research to overcome those limitations.

7.2 Data

In the future, two different methods of collecting data could be explored. If the interruption in production is allowed, the best method would be to send in the monos that are part of a mix to the AOI without performing the physical mix. This helps to uncover the order of occurrence of the links at the AOI. A decision can then be taken to capture images from all the monos uniformly, which not only helps solve the issue with data distribution but also now the labels would be known. With labels, the entire problem can be converted to a supervised classification one rather than unsupervised clustering.

If the first recommendation is still not feasible due to the interruption to production, different ways to improve current clustering performances could be explored. One way is to collect images from an entire mix, which results in about 200,000 images. With this, at least the data distribution would now be known, though the labels would still be unknown. Then the feature extraction and clustering algorithms can be trained on an entire mix to obtain more conclusive clustering results. Though the validation data could still add some noise due to the manual procedure used in collecting their images. A more automated approach could be thought about here.

7.3 Infrastructure

Both the recommendations for data includes significant infrastructure change or upgrades. The first recommendation involves altering the procedure of sending the monos into the AOI. For this, a provision needs to be made to be able to run a production line only to collect data in this recommended manner. And depending on the chosen number of samples distributed across each monos, the system should be capable of storing images for these images with the mono label attached.

The second recommendation would need the system to be able to capture and store 200,000 images. This means a significant upgrade of processing and storage space in the AOI. Also, to be able to train models with such a large dataset, would need a significant upgrade in the Deepcube's processing and storage capacity too.

Bibliography

[1] Amir Ahmad and Shehroz S Khan. Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access*, 7:31883–31902, 2019.

[2] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

[3] Dongdong Chen, Jiancheng Lv, and Yi Zhang. Unsupervised multi-manifold clustering by learning deep representation. In *Workshops at the thirty-first AAAI conference on artificial intelligence*, 2017.

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[5] Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

[6] D.Gulyas. Herleiden schakel naar charge. 2020. run:../myFolder/TbP-QMM-Int-W0557 Herleiden schakel naar charge.pdf.

[7] Theodoros Georgiou, Yu Liu, Wei Chen, and Michael Lew. A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision. *International Journal of Multimedia Information Retrieval*, pages 1–36, 2019.

[8] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 5736–5745, 2017.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[10] Warith Harchaoui, Pierre-Alexandre Mattei, and Charles Bouveyron. Deep adversarial gaussian mixture auto-encoder for clustering. 2017.

[11] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[12] Peihao Huang, Yan Huang, Wei Wang, and Liang Wang. Deep embedding network for clustering. In *2014 22nd International conference on pattern recognition*, pages 1532–1537. IEEE, 2014.

[13] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, pages 24–33, 2017.

[14] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[17] Rekhil M Kumar and K Sreekumar. A survey on image feature descriptors. *Int J Comput Sci Inf Technol*, 5:7668–7673, 2014.

[18] Yali Li, Shengjin Wang, Qi Tian, and Xiaoqing Ding. A survey of recent advances in visual feature detection. *Neurocomputing*, 149:736–751, 2015.

[19] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[20] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[21] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[22] J Alison Noble. From inspection to process understanding and monitoring: a view on computer vision in manufacturing. *Image and vision computing*, 13(3): 197–214, 1995.

[23] Kelly OBrien and Jacqueline Humphries. Object detection using convolutional neural networks for smart manufacturing vision systems in the medical devices sector. *Procedia Manufacturing*, 38:142–147, 2019.

[24] Niall OMahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

[25] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105, 2004.

[26] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[27] Sohil Atul Shah and Vladlen Koltun. Robust continuous clustering. *Proceedings of the National Academy of Sciences*, 114(37):9814–9819, 2017.

[28] Sohil Atul Shah and Vladlen Koltun. Deep continuous clustering. *arXiv preprint arXiv:1803.01449*, 2018.

[29] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[30] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[31] Javier Villalba-Diez, Daniel Schmidt, Roman Gevers, Joaquín Ordieres-Meré, Martin Buchwitz, and Wanja Wellbrock. Deep learning for industrial computer vision quality control in the printing industry 4.0. *Sensors*, 19(18):3987, 2019.

[32] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[33] Masahiko Yachida and Saburo Tsuji. Industrial computer vision in japan. *Computer*, (5):50–63, 1980.

[34] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.

[35] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[36] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. Deep clustering by gaussian mixture variational autoencoders with graph embedding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6440–6449, 2019.

Appendices

Appendix A

Process

A.1 CVT

Bosch Transmission Technology produces and assembles push belts for CVT which are found in cars equipped with automatic gearboxes. CVT is a type of automatic transmission system that is used in automobiles. A conventional automatic transmission system provides limited gear options with fixed steps between them, whereas the CVT is very flexible making the engine operate constantly while offering varied speeds for the vehicle. This is made possible by the two pulleys used by the CVT which are connected by a belt. One of the pulleys is attached to the engine which is the source of the energy for transmission, hence is also called the input pulley or the driver pulley. The other pulley isn't itself connected to any engine but is tuned by this input pulley, hence is called the output pulley or the driven pulley. The radius of these pulleys is variable relative to one another, such that, when the radius of one pulley increases the other one decreases to always keep the belt tight. This mechanism facilitates an infinite number of gears as opposed to a fixed set up in a conventional automatic transmission system. Thus continuously variable transmission provides great flexibility while the engine still always runs under optimal operating conditions. Therefore, fuel consumption and CO2 emissions are reduced.

A push belt is a key component in CVT. They are made up of steel loops and links (also known as links). Each push belt consists of hundreds of these links which are held together tightly by several steel sheets called loops. The links also come in varied sizes and thickness. They have to be manufactured with a lot of precision because any defect in them can lead to degradation of the push belt during the performance in the field. Bosch has three functional units MSE1, MSE2, and MSE3 each responsible for their function in production. A detailed description of the production cycle is provided below.

A.2 Production of Links

A group in Bosch called MSE1 is responsible for producing the individual links for a push belt. The links come in two sizes 30/12 and 24/12. A steel coil is first unrolled and fed into a fine blanking machine. Fine blanking is a process where the coil is unrolled and links of the desired shape and size are stamped out of the raw coil. These links are placed in metal containers called Mono.

After unroll and stamping the links undergo a series of procedures starting from Washing and Hardening. Here the links are heated to a certain temperature to harden them and then they go through a preliminary washing procedure. After this, they undergo a process called Body Drumbling and Deburring, where the rough parts such as the edges and corners are smoothed out using several small stones. Next, the links go through a Body grinding procedure, where further softening of uneven edges and corners is performed to smooth them out. Now, the monos have reached the End of Line and are stored in the inventory.

A.3 AOI

Here, at this stage, each mono that is stored in the inventory will have a unique identifier attached to them. This identification also reveals information such as the Fine Blanking number, track number, run number, and the date. In MSE3, these monos have to be picked up for a procedure called Mixing. Mixing is a process where different monos in the inventory are matched based on specific parameters such as the saddle height, element thickness, and more. Then the links in these monos are physically mixed. The typical size of a mix contains links from 10 different monos. After mixing, the links are put back to the same number of containers that went into mixing. Hence, the result of mixing will also be the same number of monos albeit with mixed links. After mixing the links go through another round of Washing.

Once these mixed monos are received they are sent into the Automated Optical Inspection (AOI) unit. Here individual links go through a series of tests such as the RNA tremble filter and Eddy current test. Then, the links are fed into a loop or a track which consists of a chain of six stations. Each station here consists of multiple cameras that are positioned in such a way to capture images of various parts of these links. Finally, these links pass through a final inspection called the Torsie Kop. During any of these tests, if certain links do not meet the standards, they are immediately removed from

the chain and are marked as a defected element.

After the AOI, the next and final procedure is the assembly. Here, around 100 links and a loop from MSE2 are used to assemble a single Push belt. The Push belts are then laser checked and weighed before packaging.

A.4 Manual Traceability Steps

Whenever a broken pushbelt arrives the following steps are first performed to match the optical characteristics:

1. **Optical characteristics**

   (a) The mix ID of the belt is known and is used to retrieve the mono IDs in the mix.

   (b) Using the mix ID, the mix monsters (mix bag) for the mix are obtained. Mix monsters are nothing but a small number of link samples from a mix. Here, samples from each of the 10 monos of that mix are placed in separate bags.

   (c) The number of mono bags in the mix monster is cross-verified to match the number of mono IDs retrieved in step a.

   (d) The samples are arranged in a sequence according to the list from step a. 5 links from each mono bag are taken and arranged in front of the bag and numbered 1 to 5.

   (e) The link to be matched is marked such that it doesn't mix with the samples and then their optical characteristics such as strongly repetitive patterns in punch grooves, stair radius profile are matched.

   (f) Specific patterns that are tool-related, such as punch grooves, indentations and abriss are examined on the saddle surface and the head.

   (g) Then some of the other regions of the link are examined, such as the stud and pit area.

   (h) Examinations are also conducted at both the flanks to see whether the curing and deburring processes leave traces that are monospecific.

   (i) Finally, the bottom of the links are examined for the to see patterns due to the cutting edges.

   (j) The score from the optical inspection is cross-verified by having a double-check with another colleague and then by going for a geometric inspection.

2. **Geometric Characteristics**

   These are some additional steps to be performed if the optical characteristics inspection is inconclusive.

   (a) Thickness of various regions of the link is measured and recorded using a Marameter.

   (b) These measurements are then compared with the measurements of the sample series and are recorded as a positive or a negative match against them.

   (c) Up until this point, if the match is still not established, the measured values are used to give a rating with a scale of 1 to 10 (10 is a match with 100% certainty). This rating is also rarely possible to obtain a 100% match as the measurement values will not often exactly match the sample series.

   (d) Finally, a decision will be taken whether to accept the match or take follow-up steps. [6]

Appendix B

Literature Survey

B.1 Additional Deep Clustering Techniques

B.1.1 Auto-Encoder Based

- **Deep Clustering Network(DCN)**
  Extracting features using Deep Neural Network (DNN) is usually seen as a pre-processing step for clustering. To emphasize the need for DNN, [34] suggests that, while simple linear transformation works in many cases, non-linear mapping captures complex representations better. Usually feature learning through DNN and clustering are treated in isolation as two separate entities running sequentially, with an assumption that the latent representations will be naturally suitable for clustering. But with no clustering-promoting objective extensively incorporated in the learning process, the DNN does not necessarily output features that are suitable for clustering. Hence, the paper introduces an auto-encoder based method with k-means clustering and discusses a possible collaboration in fine-tuning the reconstruction and clustering loss together through a joint optimization criterion. This way, a clustering-friendly representation can be learnt, by also making sure that the obtained features are not corrupt. In other words, reconstruction through the decoder block is critical to avoid trivial features.

  In practice, the encoder block learns the compressed representations, while the decoder reconstructs the data. The intermediate output from the encoder is applied to the k-means algorithm to obtain the intermediate clusters. An alternating optimization is applied where the reconstruction loss is tuned in the first step keeping the cluster centroids and assignments constant. In the next step, the reconstruction loss is kept constant, while the cluster centroids and subsequently the cluster assignments are updated. This way, the joint optimization criterion

can be achieved.

In comparison to other methods, such as DEC and SAE+K-means, DCN proves to map clustering-friendly representations for both synthetic and real-world datasets. DCN also proves to work quite well with all non-linear activation functions, such as sigmoid, ReLu, and tanh. For working with high-dimensional data such as images and texts, DCN is also flexible enough to be modified to a CNN. Other clustering algorithms like DBSCAN and spectral clustering can also replace k-means.

- **Deep Embedded Network(DEN)**
  Existing methods mainly focus on clustering based on the similarities/dissimilarities in data rather than extracting effective representations, criticizes [12], arguing that they affect the clustering performance. This paper talks about multiple-level features where high-level features are made up of low-level ones, arguing that hierarchical feature extraction is necessary. Hence, the paper proposes an auto-encoder based DEN which can help to solve this by imposing two constraints to learn clustering-oriented representations.

  After learning a latent representation of the original data using a deep auto-encoder, which minimizes the reconstruction error. First, a locality preserving constraint is applied to uncover the intrinsic manifold from the learned representations. This constraint is important to preserve the local structure-property of the original data. Second, a group sparsity constraint is applied, which aims to learn block diagonal representations. In these block diagonal representations, the non-zero value groups of the representations correspond to its cluster. Due to these two constraints, there is some similarity of this technique with spectral clustering. The notable difference being the introduction of non-linearity from deep learning. For pre-training, a method from [11] is adopted.

  Results indicate DEN performs better when compared to k-means and spectral clustering.

- **Deep Subspace Clustering Networks (DSC-Nets)**
  Pointing out that the most recent work on subspace clustering deals with linear subspaces, [13] argues that for image clustering, data don't need to conform to linear subspace models, thus emphasizing that the non-linearity in data needs to be captured. There are some kernel-based non-linear models, but the selection of kernel here is empirical. Hence the paper proposes an auto-encoder

based model called DSC-Nets, by introducing a self-expressive layer to harness the self-expressiveness property of a union of subspaces. This model can learn an explicit non-linear mapping of data, in an unsupervised manner, that is well-adapted for subspace clustering.

The self-expressive layer is present between the encoder and the decoder. At first, the deep auto-encoder is pre-trained without the self-expressive layer. Then the entire network is fine-tuned using the pre-trained model for initialization. It is a fully-connected network without bias and non-linear activations. This network encodes the self-expressiveness property of data drawn from a union of subspaces, that is, each data sample is represented as a linear combination of other samples in the same subspace. This helps to learn affinities between all data points through their combination coefficients. The paper gives two steps in subspace clustering. First, to extract the affinity matrix, which estimates the affinity of a pair of data points. Second, to apply normalized cuts [29] or spectral clustering [21].

The paper concludes that self-expressiveness is more robust to outliers and noise. Finally, experiments on four standard image datasets had demonstrated that the DSC-Nets provides a significant improvement over the state-of-the-art subspace clustering solutions, in terms of clustering accuracy. However, the major drawback is that, due to the high complexity of the self-expressive network, it requires high-processing power for large data sets.

- **Deep Multi-Manifold Clustering (DMC)**
  In multi-manifold clustering (MMC), the learned representations are said to be meaningful only if the local characteristics of data are preserved. Meaning, each data point can be recovered using the representation of its nearby point than using that of a non-nearby point. To solve this objective, [3] proposes a auto-encoder based framework called DMC.

  In addition to the reconstruction loss from the auto-encoder, the network tries to iteratively minimize a joint cost function. First, a locality preserving objective is minimized to iteratively explore data relations and learn structure-preserving representations. Second, a clustering-oriented constraint is applied to extract both cluster-specific and discriminative representations.

  Experiments are conducted to compare the results of DMC against nine different

state-of-the-art MMC methods on three widely used datasets. It is observed that DMC achieves higher clustering accuracy than all the other compared methods on all the three datasets. In comparison to some unsupervised MMC methods, it is proven that DMC can even extract more precise object-specific information. When tested for its structure-preserving capability, DMC again outperforms its competitors. Finally, the paper also substantially proves that deep architectures with at least two hidden layers always extract more meaningful information than traditional methods.

- **Deep Embedded Regularized Clustering (DEPICT)**
  As discussed, currently there are a lot of dimensionality reduction techniques being used to project data into low-dimensional spaces. However, [8] cites four issues with existing clustering algorithms while dealing with real-world image data.

  1. Using inflexible hand-crafted features, which do not depend on the input data distribution;
  2. Using shallow and linear embedding functions, which are not able to capture the non-linear nature of data;
  3. Non-joint embedding and clustering processes, which do not result in an optimal embedding subspace for clustering;
  4. Complicated clustering algorithms that require tuning the hyper-parameters using labeled data, which is not feasible in real-world clustering tasks. [8]

  To address these issues, a new auto-encoder based technique has been proposed called DEPICT. In addition to the reconstruction loss from the auto-encoder, the model is trained alternatively with two other objective functions in a joint cost optimization. First, a multinomial logistic regression (soft-max) stacked on top of a multi-layer auto-encoder. The encoder and the softmax layer are trained using the relative entropy (KL divergence) minimization for discriminative clustering. Second, a regularization term is added based on a prior distribution to restrict the frequency of cluster assignments. The regularization term penalizes unbalanced cluster assignments and prevents allocating clusters to outlier samples. During a standard learning approach, the reconstruction error is first used to pre-train the encoder and decoder parameters, and the encoder parameters are then fine-tuned using the two objective functions previously discussed. However, fine-tuning without the reconstruction loss may overwrite the encoder parameters entirely, which could consequently negate the benefit of the layer-wise pre-training step. Hence to avoid this problem, all of the encoder and decoder

layers together along with the soft-max layer undergo simultaneous training, by entirely discarding the pre-training step.

Based on experiments on six benchmark datasets against several state-of-the-art methods, it has been observed that the clustering performance using this model is better in almost all cases. Though JULE [35] performs slightly better in some cases, the running times increase exponentially with scale, while increasing linearly for DEPICT. Hence, this approach gives superior results in comparison to other alternatives, while also providing faster running speeds.

- **Deep Continuous Clustering (DCC)** Most of the clustering algorithms require setting the number of clusters a priori. The usual optimization procedures involve discrete re-configurations of the objective. For example, the data points are discretely reassigned to centroids or in Agglomerative clustering, the procedure of merging clusters is also discrete. Thus it is challenging to integrate them with an optimization procedure that modifies the embedding of the data itself [28].

  This paper proposes an auto-encoder based model with three objectives to be fulfilled. First, the joint problem of these three objectives needs to be expressed as an optimization of a single continuous objective. Second, this optimization should be compliant with scalable gradient-based solvers such as modern variants of SGD. Third, since the number is often not known in advance, the model should not require setting the number of clusters a priori. The initializing of network parameters is performed using a stacked denoising auto-encoder (SDAE). Each pair of corresponding are pre-trained sequentially from the outer to the inner. After all layer pairs are pre-trained, the entire SDAE is fine-tuned end-to-end using the reconstruction loss.

  For fully connected networks, DCC performs far better than DCN and DEC. For convolution networks, this approach performs better than DEPICT and JULE on certain data sets. The paper also proves the importance of joint optimizations by comparing DCC against applying SDAE on K-means and DBSCAN. When the representations learned through DCC are applied on k-means and DBSCAN, the clusters automatically improve as well.

  An application of DCC based on robust estimation is described in [27] as Robust Continuous Clustering (RCC). The paper describes two RCC objectives. The first objective constrains the representatives to remain near the corresponding

data points. The second objective pulls the representatives to each other, encouraging them to merge. These along with the reconstruction objective of the auto-encoder form the joint cost function.

B.1.2 Variational Auto-Encoder Based

- **Variational Deep Embedding (VaDE)**
  Utilizing the advantages of Gaussian distributions at the bottleneck in VAEs, [14] introduced VaDE which combines clustering with VAE. As already highlighted, it is important to learn good representations to obtain better clusters. VaDE models the data generative process with a DNN and GMM. Here, a single Gaussian prior of the VAE is replaced by a Mixture-of-Gaussians prior, making VaDEs more suitable for clustering.

  The process of VaDE firstly picks a cluster by the GMM. From this Gaussian distribution, a latent representation is randomly sampled. This latent representation is then fed to the decoder to obtain the input reconstruction. Then, the reconstructed input is fed to the encoder to learn the latent embedding. The overall optimization is performed by maximizing the Evidence Lower Bound (ELBO) using the Stochastic Gradient Variational Bayes (SGVB) and reparameterization trick.

  An evaluation of performance was conducted on five benchmark datasets comparing VaDE against several architectures, such as, AE+GMM, DEC and VAE+GMM. In terms of clustering, the experiments prove that VaDE provides state-of-the-art performances for all datasets. In terms of generation of samples, VaDE can generate highly realistic samples for any specified cluster, without using any supervised information during training.

- **Gaussian Mixture VAE (GMVAE)**
  Similar to VaDEs, [36] also proposes a variant of VAE with Gaussian mixture as a prior. The paper postulates that generative models such as VAE can be applied for unsupervised clustering. Based on an assumption that data generation happens from a multi-modal prior distribution in the latent space, where optimization can be done using a reparameterization trick. Additionally, the paper mentions a problem of over-optimization in VAEs which can severely affect the performance of clustering. The paper further argues that, with a Gaussian prior, though latent representations are disentangled and structured, due to its uni-modal structure it

is ineffective against complex representations. Hence, GMVAE has a Mixture of Gaussians prior which is multi-modal.

With an assumption that inferring the class of a data point is equivalent to inferring which mode of the latent distribution the data point was generated from, the latent space can thus be segregated into distinct classes or clusters. However, this inference is non-trivial. To apply the reparameterization trick on discrete variables, an architectural adjustment of VAE is made for optimizing ELBO. Further, the problem of over-optimization of VAE can be overcome with an additional constraint called as the minimum information constraint, based on an improved approach proposed in [16].

A study is conducted on the inference process in a low-dimensional synthetic dataset with a focus particularly on the over-regularisation problem's effects on the clustering performance of GMVAE. Then an evaluation is conducted on state-of-the-art datasets for unsupervised clustering. It is proved that the over-regularization problem can be solved with standard heuristics. Also, the clusters obtained are observed to be meaningful and images generated from them are observed to share relevant high-level features with the latent space.

B.1.3 Generative Adversarial Network Based

- **Deep Adversarial Clustering (DAC)**
  A deep generative model based on Adversarial Auto-Encoders (AAE) is proposed in [10] called DAC. An auto-encoder is used to learn latent representations and a Gaussian Mixture Model (GMM) is used for clustering task.

  First, a reconstruction cost is introduced by the auto-encoder for learning good latent representations. Second, an adversarial part is added with a GMM. Lastly, a discriminator is used to force the latent space to follow the Gaussian mixture prior. Hence, results in an objective function with three terms, the traditional auto-encoder reconstruction objective, the Gaussian mixture model likelihood, and an adversarial objective.

  Based on experiments several benchmark datasets, results show that DAC performs better than state-of-the-art methods. There is an extra boost inaccuracy from Ensemble Clustering that combines multiple outputs coming from multiple

random initializations. The results then show that an auto-encoder dramatically improves all clustering results. Also, the adversarial contribution outperforms all previous variants.

- **Categorical Generative Adversarial Network (CatGAN)**
  The generative objective of GANs using the reconstruction loss and the discriminative objective of a classifier are combined to form a Categorical Generative Adversarial Networks (CatGAN) for unsupervised and semi-supervised learning [30]. For an unsupervised setting, the CatGAN can be obtained by generalizing GAN framework to multiple classes. The proposed model modifies the current functioning of the discriminator, which segregates its input into two classes, "real or "fake". Instead, the discriminator is asked to classify the input into multiple classes. Although still staying uncertain of the class assignments for samples from the generative model, making the model more robust. In other words, the objective of the generator is not to generate samples that belong to the dataset, but to generate samples that belong to one of the K classes.

  For the proposed model to work, there are some requirements for both discriminator and generator. Firstly, the discriminator must be certain of class assignments of known samples and be uncertain of class assignments for generated samples and treat all classes equally. Then, the generator must generate samples that are equally distributed over all classes and each sample is highly certain of a class assignment. It is well known that GANs are hard to train. The discriminator can learn too quickly making the loss for the generator saturate. Also, the generator might get stuck generating only one mode of data or wildly switching between generating different modes during training. Batch normalization with the introduction of Gaussian noise to hidden activations is found to have helped overcome this problem.

  Empirical results show that CatGANs are competitive to state-of-the-art methods on almost all datasets. In terms of clustering performance too they prove to perform better than K-means and RIM, especially on a circles dataset. Also, a semi-supervised CatGAN is said to extract better representations than RIM and GAN.

- **Information Maximizing Generative Adversarial Network (InfoGAN)**
  In contrast to other approaches previously discussed, [4] proposes a method named InfoGAN which learns disentangled representations. They independently encode the salient attributes of data. The method maximizes the mutual informa-

tion between latent representations and observation. For clustering tasks, there is a major research focus on generative models with an expectation that the ability to synthesize data entails learning good disentangled representations.

The method introduces a mutual information cost that penalizes a lack of mutual information between a fixed small subset of the GANs noise variables and the observations. This loss helps learn interpretable and meaningful representations that could prove crucial for the clustering task. Due to its simple nature InfoGANs add only negligible computational cost and comes free with GAN. The input vector is decomposed into two parts namely, incompressible noise vector and latent code. To avoid trivial latent codes, an information-theoretic regularization is used to ensure that the mutual information between latent codes and generator distribution is high.

The method disentangles both discrete and continuous latent variables and can also scale-up to complicated datasets. Experiments were conducted to prove two factors. First, to investigate whether mutual information can be maximized efficiently. Second, to evaluate if InfoGAN can learn disentangled and interpretable representations. Both these theories were conclusively proved for an unsupervised approach on benchmark datasets.

B.1.4 Discussion

The literature survey provided useful insights into different types of clustering and deep clustering methods. The findings from the literature and the initial results, encourage to look beyond traditional techniques and emphasizes the need to investigate into deep clustering. Hence, this section aims to provide useful observations about different deep clustering architectures learned from literature. In literature, deep clustering methods are classified into four categories based on their underlying architecture.

- Auto-Encoder based
- Variational Auto-Encoder based
- Generative Adversarial Network based

Auto-encoders based methods can be coupled with most of the clustering algorithms, where both the clustering loss and reconstruction loss are jointly optimized.

All of these methods perform better than most of the existing methods, such as k-means. DCN and DEC are some of the basic versions of AE based deep clustering. Each method aims to learn better representations by also jointly optimizing the clustering loss. Specifically, DMC specializes in multi-manifold clustering problems. It is observed that all these methods provide good clustering results and perform well in terms of prediction accuracy. In contrast, DSC-Nets suffers from performance issues due to a complicated middle layer that is newly introduced. This layer aims to make the model more robust to outliers and noise. DEPICT performs better for large-scale datasets as its complexity only increases linearly with scale. A fully-connected DCC is said to be more accurate and consistent than a fully-connected DCN or DEN and certainly performs better on high-dimensional image datasets. Among deep clustering methods using convolutional networks, the performance of DCC is found to be better than DEPICT on some datasets like COIL100 and YTF. A summary of comparisons, for all these auto-encoder based deep clustering methods, can also be found in Table B.1.

Deep-generative models VAE and GAN are very computationally expensive because of their generative nature and the structural changes they introduce at the bottleneck. Unlike auto-encoders, for VAEs, the clustering loss is integrated within the model. The two proposed VAE based models in literature, VaDE and GMVAE, are quite similar to each other. The only difference is that GMVAE adds a minimum information constraint to regulate VAE's over-optimization problem. Due to this, they are complex to implement and computationally expensive when compared to VaDE. VaDE on the other hand, are based on the basic VAE architecture. Hence, they don't add much computational cost to the existing cost of VAEs.

Finally, some GAN based methods, such as InfoGAN, CatGAN, and DAC were also introduced earlier. It is understood that all three methods perform quite well in both their generative and clustering tasks. DAC especially has a simple and easy to use architecture. Also, the results are quite encouraging. Interestingly, though CatGAN gives good clustering performances, they are known only to extract better representations with semi-supervised learning. But the method that stands out is InfoGAN, for its ability to learn disentangled representations. They also try to maximize the mutual information between latent code and distributions, without adding much overhead to GANs.

Comparing the methods from four major techniques against each other, Auto-encoders are known to learn good latent representations due to their reconstruction error minimization approach. But generative models are also capable of generating

new samples rather than just reconstruction. To achieve random sampling, they are known to limit latent distributions by introducing constraints in the latent space. However, due to their generative capability, their computational complexity also is very high in comparison to vanilla auto-encoders. A summary of comparisons for all four deep clustering categories is shown in Table B.2.

Method	Specialties	Objective Functions
DCN	1. Conventional and Simple 2. No additional add-ons 3. Just basic auto-encoder architecture with clustering algorithm	1. Reconstruction Loss 2. Clustering Loss
DEN	1. Adds a locality preserving constraint - to retain local structures 2. Adds a group sparsity constraint - to assist in clustering	1. Reconstruction Loss 2. Locality-preserving Loss 3. Group Sparsity Loss
DSC-Nets	1. Introduces self-expressiveness 2. As a result a self-expressive layer is added at the bottleneck	1. Reconstruction Loss 2. Self-expressiveness
DMC	1. Applicable for multi-manifold clustering 2. Adds a locality preserving constraint - to retain local structures 3. Adds a clustering-oriented constraint - to form better clusters	1. Reconstruction Loss 2. Locality-preserving Loss 3. Clustering-oriented Loss
DEPICT	1. Performs better even on large datasets 2. Adds a multinomial softmax function on the top-layer 3. Adds a regularization term - to restrict unbalanced clusters	1. Reconstruction Loss 2. Multinomial Classification 3. Balanced Clustering
DCC	1. Tries to replace discrete nature of clustering 2. Proposes the joint objective to be a single continuous objective 3. Proposes the model to be compliant to modern variants of SGD 4. The model does not require setting the number of clusters	1. Single continuous objective

Table B.1: Auto-Encoder based methods comparison

Category	Specialties	Object Functions
Auto-Encoder	1. Joint optimization of two or more objectives 2. No trivial solutions 3. Feature space is not corrupt	1. Reconstruction Loss 2. Clustering Loss
Variational Auto-Encoder	1. Can also generate new samples 2. Imposes a GMM priori to VAE 3. High computational-complexity	1. Reconstruction Loss 2. Clustering Loss
Generative Adversarial Network	1. Can also generate new samples 2. Imposes a multi-class priori to GAN 3. Possibility of a mode collapse 4. High computational-complexity	1. Reconstruction Loss 2. Clustering Loss

Table B.2: Deep clustering categories comparison

Appendix C

Additional Results

As an initial study, experiments were conducted using a small set of labelled data to investigate whether there is an influence of process or the tool wear on the grooves. A detailed description of the feasibility dataset is provided in section C.1.1. As mentioned there, the dataset includes images for links from both FB and EOL locations. All the images also come with labels such as their run ID, track ID and FB ID. Though relatively a small-sized dataset it can still provide good insights into the future scope of this research.

C.1 Feasibility Study

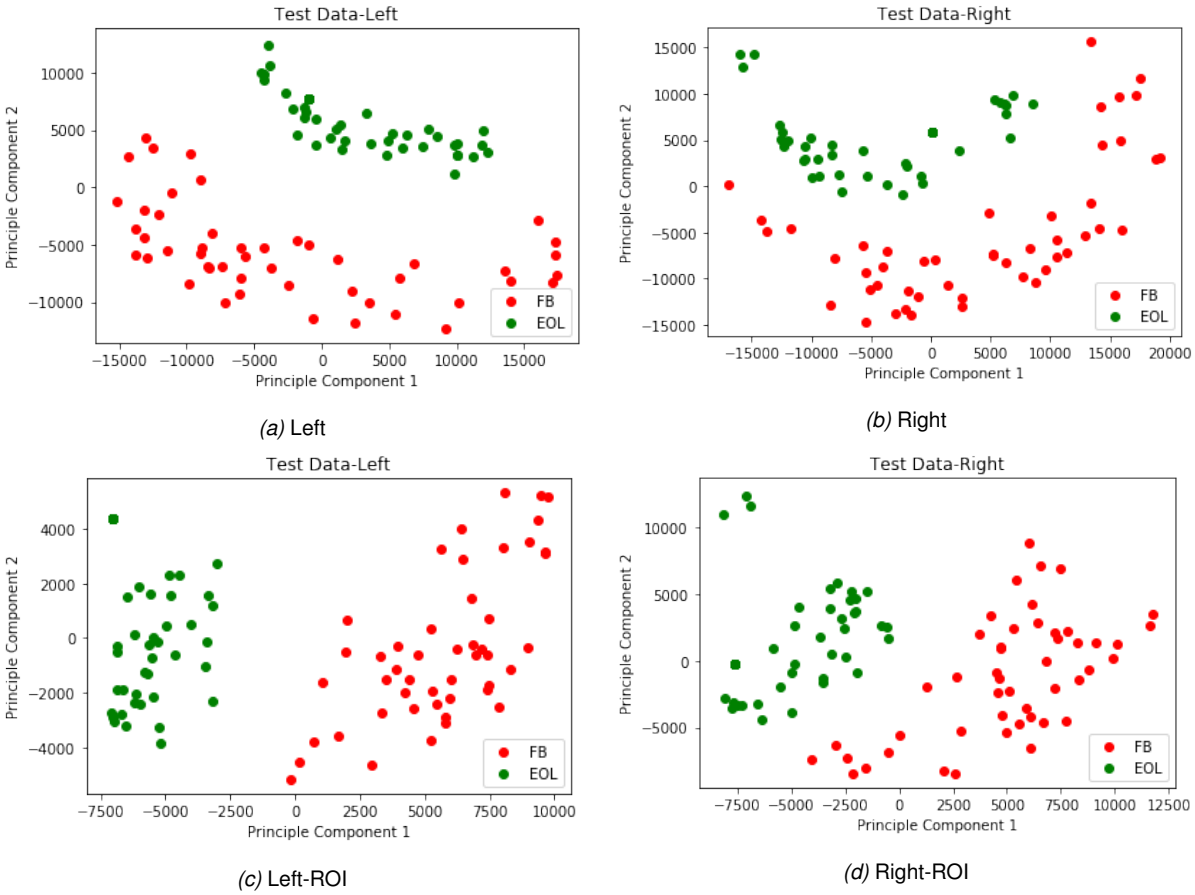
C.1.1 Feasibility Data

The feasibility test which will be explained in the further section was a test that used a small set of labelled data. The samples were collected such that, a single FB machine was allowed to complete four different runs, where the first run was with a clean and refurbished combination of the cutting plate, cutter and ejector. This resulted in 16 monos in total as each FB machine also has four tracks. Since the identity of these monos gets lost after mixing, to retain their labels it is important to collect the samples from a stage called EOL which is just before. 3 samples each were collected at the EOL and an additional 3 samples each were collected after FB from the same monos. This additional step was necessary to verify the influence of process since after FB the links go through a chain of processes before reaching Mixing. And there is a possibility of the grooves being modified by one of the intermediate processes in-between. Hence, the total resulting sample size would be 96 links. Images for these 96 links are taken manually at AOI Station 6. As station 6 gives images from both the orientations for 96 samples, a total of 192 labelled images were collected.

C.1.2 Feasibility App

A UI application called the Feasibility App was developed using HTML and Flask in Python, using which a user can visually inspect these images. The application provides users with a basic filter feature that allows them to select the two samples that they desire to compare. Besides, there is also an advanced filter that allows the users to select a single sample and make a comparison against samples across others runs or tracks. The comparisons are based on several distances and correlation metrics offered by OpenCV and Scipy. The metrics are applied to the histograms of these images. Besides, there is a third filter option that provides some basic 2D visualization of data using PCA and t-SNE. Finally, the app allows users to perform all the analysis on both raw images or only on the region of interest.

C.1.3 Results



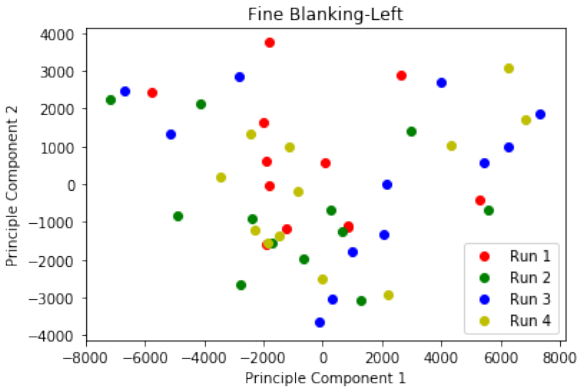
(a) Left

(b) Right

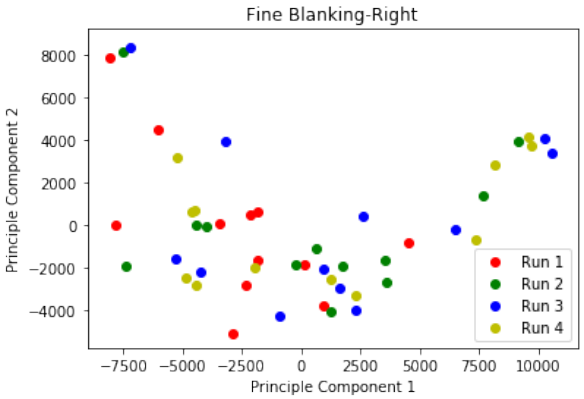
(c) Left-ROI

(d) Right-ROI

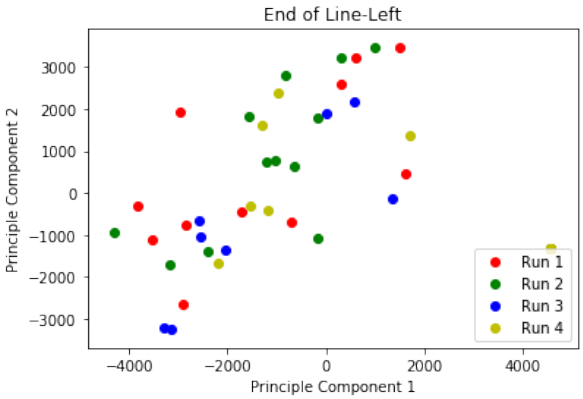
Figure C.1: A graphical representation of the points to visualize the influence of process on the grooves using PCA (a & b) All images of the left and right saddle surface before pre-processing (c & d) All images of the left and right saddle surface after pre-processing



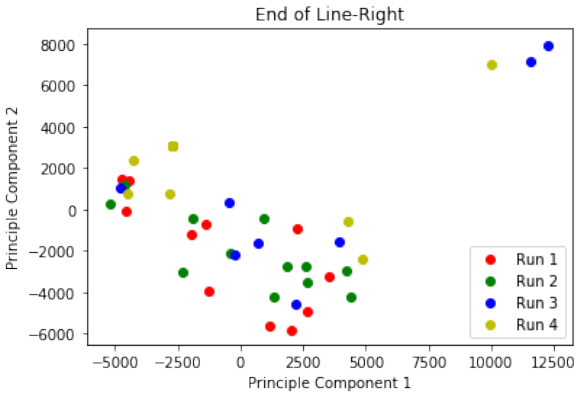
(a) FB-Left



(b) FB-Right



(c) EOL-Left



(d) EOL-Right

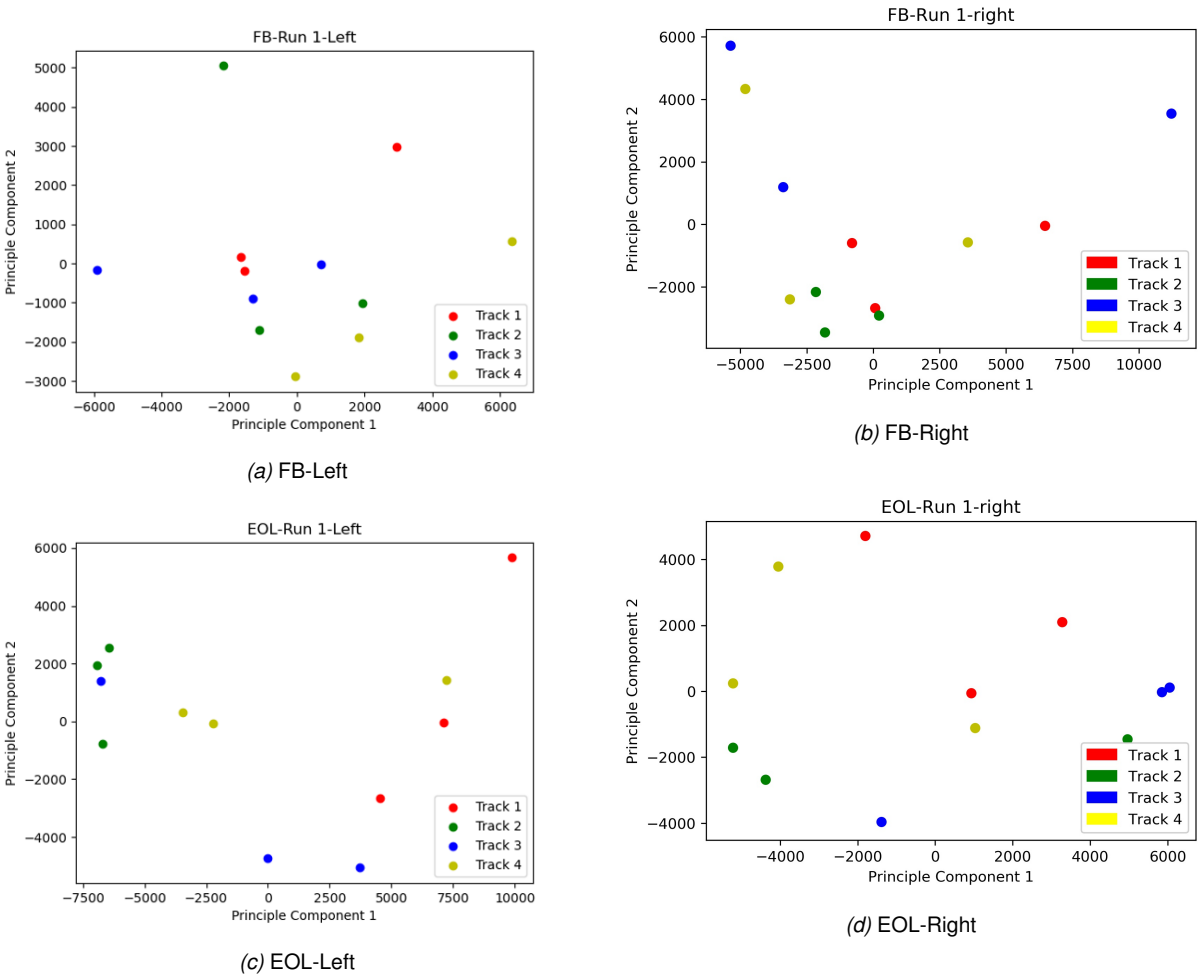
Figure C.2: A graphical representation of the points to visualize the influence of runs on the grooves using PCA (a & b) All images captured at the FB for the left and right side of saddle surface (c & d) All images captured at the EOL for the left and right side of saddle surface

Some of the initial results obtained from the feasibility study introduced in section **??** will be presented here. The study is designed to apply some basic unsupervised computer vision feature extraction and clustering algorithms on a small set of labelled data.

The influence of process is first captured through 2D plots seen in Figure C.1. The first two resulting principal components capture the high-variance information of the data. Here, different colours are used to indicate data points from FB and EOL. The figure includes separate plots for both the orientations, left and right and separate plots for raw images and images cropped only to the region of interest. The plots prove clear separation between points from FB and EOL.

Further, to investigate the influence of tool-ware on the grooves. In Figure C.2, separate plots are plotted for FB and EOL for both their left and right orientations to verify the influence of runs. In these plots, Run 1 indicates the first time a refurbished die-plate is being used for production. Specifically for this experiment, the die-plate is

allowed to run for four runs before it is refurbished again. That is, each die-plate is allowed to run from Run 1 to Run 4. Different colours are used to indicate different runs. It is evident from these plots that there is no visual separation of data points similar to what was seen from plots in Figure C.1.



(a) FB-Left

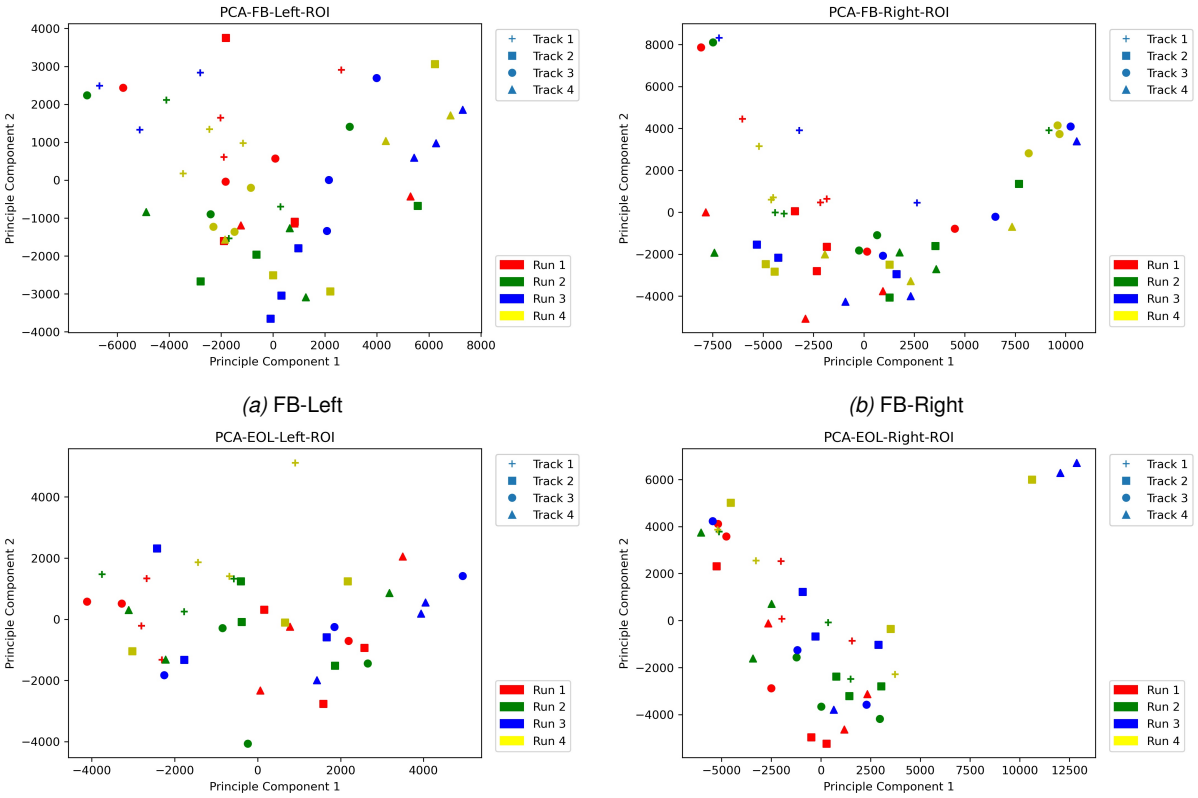
(b) FB-Right

(c) EOL-Left

(d) EOL-Right

Figure C.3: A graphical representation of the points to visualize the influence of tracks on the grooves using PCA (a & b) All images captured at the FB for the left and right side of saddle surface (c & d) All images captured at the EOL for the left and right side of saddle surface

The investigation was continued to verify the influence of tracks on the groove patterns. The plots shown in Figure C.3 provide an insight into the spatial distribution of data points for one particular run. In this case, Run #1 was taken as an example. Different colours are given to indicate points from different tracks. Each track is represented by three of its samples who all belong to the same mono. It can be observed from these plots that in some cases two out of the three points lie close to each other. This hints at the existence of some similarities between them. Though these might not be the most robust clusters, they are still better than previous results and promising enough to investigate further.



(a) FB-Left

(b) FB-Right

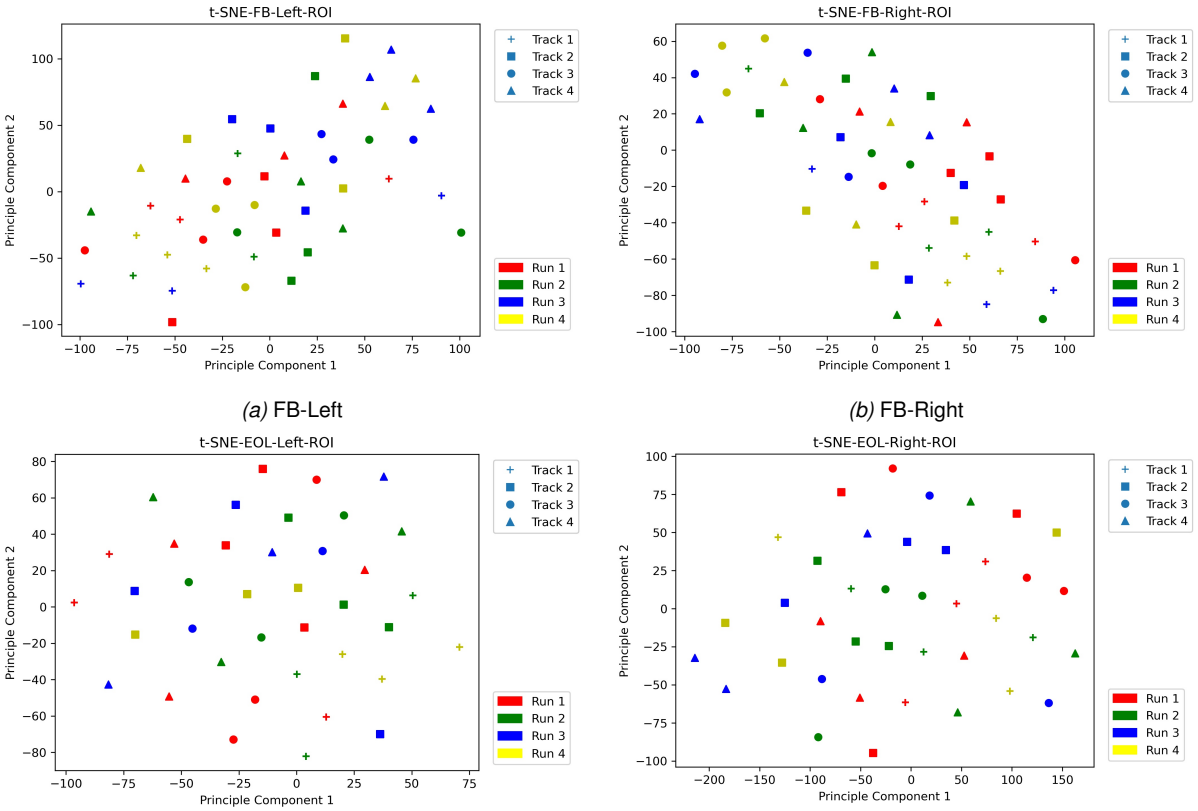
(c) EOL-Left

(d) EOL-Right

Figure C.4: A graphical representation of the points to visualize the influence of runs and tracks on the grooves using PCA (a & b) All images captured at the FB for the left and right side of saddle surface (c & d) All images captured at the EOL for the left and right side of saddle surface

For a unified visualization, both plots from Figure C.2 and Figure C.3 were combined into a single chart as seen in C.4. This plot here uses different colours to indicate points from different runs and different markers to indicate points from different tracks in a single graph. Similar to earlier figures, Figure C.4 shows separate plots for FB and EOL and for both orientations.

In addition to PCA, another method called t-SNE was used to utilize the benefits of manifold learning to verify if the non-linear method can learn better representations of data. The plots seen in Figure C.5 indicate that the results of t-SNE don't differ much from that seen in linear PCA.



(a) FB-Left

(b) FB-Right

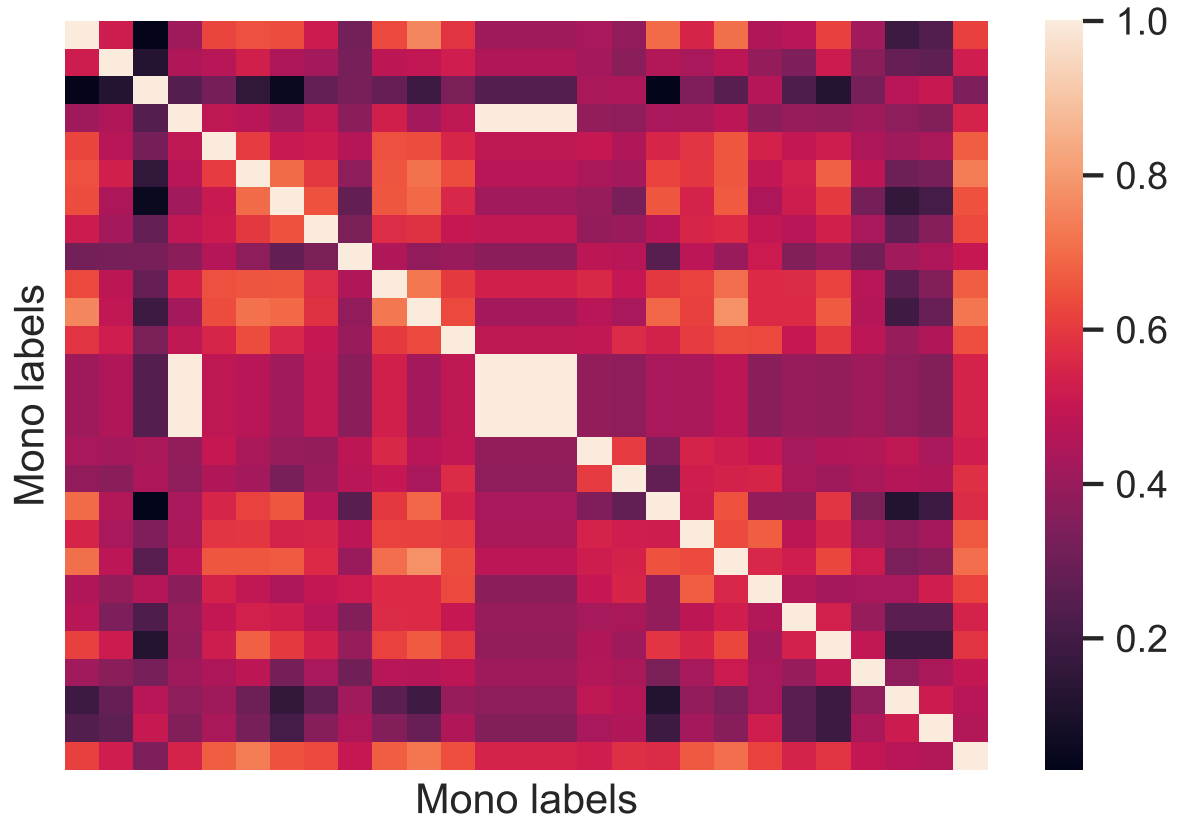
(c) EOL-Left

(d) EOL-Right

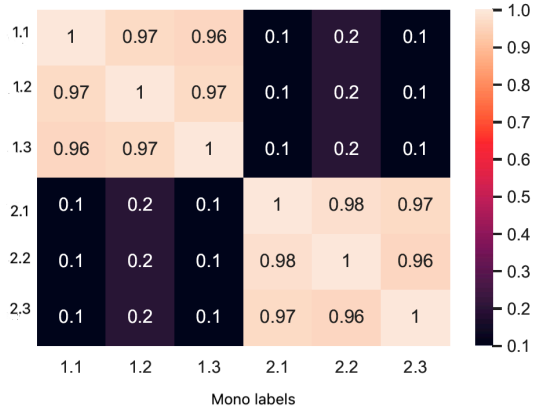
Figure C.5: A graphical representation of the points to visualize the influence of runs and tracks on the grooves using T-SNE (a & b) All images captured at the FB for the left and right side of saddle surface (c & d) All images captured at the EOL for the left and right side of saddle surface

The main inference from the study was that there is a change in the pattern structure on the saddle surface between the start and end of the line. Experiments were conducted using 2D visualizations of basic feature extraction methods PCA and T-SNE. However, it was inconclusive from this study to prove which process parameter has a greater influence on the patterns on the saddle surface. But these results motivated to broaden the research towards more complex deep learning based approaches like auto-encoder and deep clustering with a larger dataset.

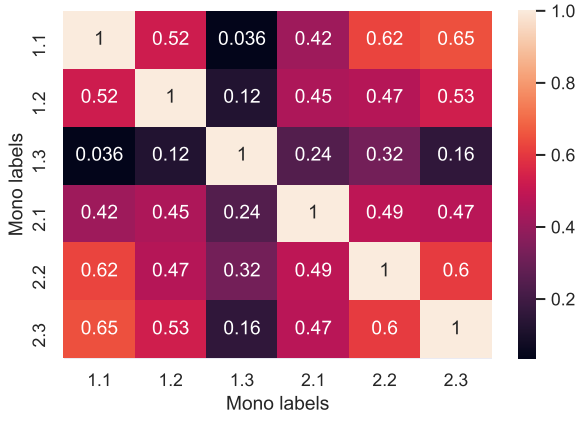
C.2 1D-Vector Correlation



(a) 1D correlation map of 27 * 27 images (9 monos) from the validation dataset



(b) Expected 1D correlation map of samples from the first 2 monos



(c) Actual 1D correlation map of samples from the first 2 monos

Figure C.6: Graphical representations of input data points converted to 1D vectors

The patterns on the saddle surface can be seen as a collection of light and dark vertical lines placed along the horizontal axis as seen in Figure 3.3. Hence, the intensities along the vertical direction are expected to have pixel values that are almost equal to

each other at any point on the horizontal axis. As an additional experiment, the pre-processed dataset which is of the size 128*1024 were converted into a 1D vector by averaging out the pixel values along the horizontal axis. The resulting vectors were of the size 1*1024.

A correlation matrix of 1D vectors of the 27 validation samples was plotted to verify if there exist similarities among samples belonging to the same mono. However, this experiment failed to prove the existence of a strong correlation between samples from the same mono. Figure C.6a shows a heat-map of the 27*27 samples from all 9 monos of the validation set. For better visualization, a smaller map consisting of samples from only the first two mono labels was plotted as shown in Figure C.6c. In this figure, the mono and sample are labelled such as 1.2, where 1 indicates the mono ID and 2 indicates the sample ID.

An expected solution would look as shown in Figure C.6b. When the 3 samples from each mono are very similar to each other, the correlation values for those pairs are expected to be closer to 1, indicated by the bright spots in the expected figure. And in contrast, when the two samples are from different monos, the correlation scores for those pairs are expected to be closer to 0, indicated by the dark patches in the same figure.