university of groningen

faculty of science and engineering

Deep Learning for Semantic Embedding and Anomaly Detection in LOFAR Data

Master's Thesis
University of Groningen

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at the University of Groningen under the supervision of
Prof. dr. L.R.B. Schomaker (Artificial Intelligence, University of Groningen)
and
Dr. Albert-Jan Boonstra (ASTRON)
and
Dr. Sarod Yatawatta (ASTRON)

Klemen Voncina
(s2900874)

August 15, 2023

Acknowledgement

I can only hope to convey in words my gratitude to Lambert, Albert-Jan and Sarod for their patience and unwavering support throughout this project. With your guidance and the encouragement of a few similarly persistant friends along the way I've learned more than I could possibly have imagined going in. On the other side of it, I am truly grateful for the knowledge you've given me, and encouraged me to find. And most importantly, for the lasting desire to keep learning both within the field of AI and the myriad of enchanting disciplines outside of it.

Abstract

This thesis explores the application of several self-supervised representation learning techniques for anomaly detection in spectral data. These techniques include five distinct methods, four deep learning-based auto-encoder models, and a linear transform method known as Independent Component Analysis (ICA). The data used in this research is obtained from an aperture synthesis radio telescope, specifically LOFAR. The primary aim is to evaluate the potential of these techniques in creating semantically meaningful embeddings. These embeddings are expected to aid in tasks such as data inspection, classification, and unsupervised anomaly detection. To achieve this aim, the methods are trained in a self-supervised manner and evaluated on a limited, labeled dataset curated for this study. One of the deep learning models tested includes a novel cascading convolutional auto-encoder architecture, uniquely adapted to account for time-frequency data characteristics and to include spatial context within its encoding. The proposed method shows best average classification scores across Random Forest and Gaussian Naive Bayes classifiers in long time-scale downsampled observations (43.83%). We demonstrate second best average accuracy in short time-scale downsampled observations (76.14%). It is beaten out in the combined embedding by ICA, however still shows itself to be the best all-round method with an average accuracy over all test conditions of 55.49%. The next best method appears to be the Sliced Wasserstein auto-encoder with an average accuracy of 53.98%. Despite introduced sparsity constraints, our method shows further learning potential even when compared to other AE based methods without overfitting, hinting at good data-efficiency properties. Additionally, this research highlights the challenge of evaluating models in a problem space with sparse labels and large variances in data samples.

Contents

Chapter 1

Introduction

As part of any data collection task, one must contend with a range of anomalous and novel events. What's more is the original definition of a domain may not include all possible anomalous states. Be it for the purpose of identifying malicious or faulty behaviour [1, 2, 3] or simply ensuring that system faults do not propagate to later stages of a processing pipeline, it is necessary to ensure that in large continuous operation systems tools are available for the identification and removal or repair of anomalous data. Anomaly detection is a paradigm in AI which deals with the identification and categorization of data samples that deviate significantly from what is expected as the normal [4]. When the negative connotations of 'anomaly' are not appropriate, the term *out of distribution detection* may be used interchangeably. Techniques for anomaly detection fall into one of the following three categories; supervised, semi-supervised and unsupervised. Supervised anomaly detection relies on labeled datasets, which are generally partitioned into normal and anomalous subsets. The task then becomes a typical classification problem. However, obtaining labeled data for anomaly detection is often a challenge due to the scarcity of anomalous examples. Semi-supervised anomaly detection, on the other hand, assumes that the training set only includes normal data. It creates a model representing this normal behavior and identifies anomalies as significant deviations from this model. One may also call this *novelty detection* [5, 6, 7]. In unsupervised anomaly detection, no labels are provided. These methods rely on the intrinsic properties of the data, and they assume that anomalies are data points that were generated by different mechanisms compared to the rest of the data. Clustering, density estimation, and subspace methods are typical techniques used for unsupervised anomaly detection.
For low-dimensional data, it is convenient to determine the statistics directly from the raw data. However, in high-dimensional data, e.g. images or spectograms, this would entail computing the statistics for each pixel. It is clear that for this kind of data, some form of dimensionality reduction is needed. Usually this is done through the process of feature extraction where some lower dimensional features are computed from higher dimensional data. This can come in the form of a linear transform or a more complex method. In much the same way as Word2Vec [8] tries to map semantic relations between words in text data, it is possible to perform a similar embedding for other forms of data.

1.1 Anomaly Detection in Radio Astronomy

The field of radio astronomy uses continuous monitoring systems to make observations in the spectral domain between 10 MHz and 738 GHz [9]. The exact methods of collection and data processing vary from system to system, but radio telescopes can be broadly divided into two major categories. Large continuous *filled-aperture* systems consisting of a single large dish, and distributed systems applying instead *aperture-synthesis* [10] to make high resolution observations. The latter carries

several benefits with regards to limitations on angular resolution at the cost of making the observation process much more complex, compute and data intensive.
The LOw Frequency ARray (LOFAR) telescope in The Netherlands, operated by the Netherlands Institute for Radio Astronomy (ASTRON) [11] is one such distributed aperture-synthesis system. This telescope is geared towards extremely low frequency observations of radio waves originating beyond the Earth. Its observations are made on the lower end of the spectrum with regards to the range of explored frequencies in radio astronomy. LOFAR operates in the frequency ranges from 20-240M Hz or as far as radio astronomy is concerned right down to the practical limit of frequencies which are able to pass through the Earth's ionosphere [12]. This observation capacity is covered by two different types of antennae: A low-band antenna and a high-band antenna. In Figure 1.1.1 both of these antenna types are visible, the high band antennae as hexagonal arrangements of tiles and the low band antennae are visible as the smaller rectangles interspersed between the tile arrays. These antenna types cover most of the aforementioned range apart from a blind spot between about 85 MHz and 108 MHz where both antennae types' sensitivity is low. Therefore, in the data as well as the physical antennae, there is a clear delineation between low band (LBA) and high band (HBA) observations.
Among a variety of applications, LOFAR has enabled the study of a previously under-studied area of early universe matter transitions, namely the Epoch of Reionization problem [13, 14] or the second matter transition period after the Big Bang. Unlike some other radio telescopes receiving in the hundreds of MHz or even GHz, LOFAR is designed to brush up right against the physical limit of frequencies that are able to pass through the Earth's ionosphere. This lower bound is found around the 20 MHz mark and below this limit radio waves are not detected because of a range of phenomena, namely scintillation, absorption and reflection [15]. Depending on precise conditions in the ionosphere the scintillation effect can be observed in higher frequencies as well. This manifests as an unexpected signal delay between two antennae. An example of this phenomenon in data recorded by LOFAR can be seen in Figure 1.1.2, the vertical bars tapering down through the frequency spectrum represent the telltale signs of scintillation. The plot also shows signs of radio frequency interference (RFI) which happens to be unavoidable in most observations. This is because most of the LOFAR observation space is also covered by assigned communication frequencies. The reason LOFAR is able to avoid most of these, however, is because there are gaps in this assignment schema where observations are able to be made [16]. Existing systems are in place to mitigate the effects of RFI on the final data product [17]. However, it is yet another example of the challenges faced by the LOFAR team in observing and categorizing anomalous behaviour in the observations produced. Even a nominally clean data sample as seen in Figure 1.1.3 can still suffer from relatively heavy RFI.
LOFAR, functioning as an aperture synthesis telescope, allows each observation to be constructed from numerous correlated baselines, corresponding to the total number of stations. This configuration can result in the generation of a significant amount of data, potentially several terabytes, for a single observation, posing challenges for manual inspection.
While the anomalies and defects are relatively easy to identify in the generated inspection plots, the motivation to automate this detection and categorization process arises from the sheer volume of data that requires processing. Unlike a filled-aperture or dish type telescope with a single receiver that can be articulated or re-positioned above a static dish, an aperture synthesis telescope employs numerous smaller telescopes that are individually pointed.
For instance, the Westerbork Synthesis Telescope utilizes a series of dishes organized in a perfect East-West line which can be steered mechanically. Conversely, LOFAR simplifies this process by pointing each station through the introduction of signal delays between adjacent antennas, a technique that is subsequently repeated at the station level. Such a processing intensive strategy has been enabled by the advancements in computational power over the past two decades.

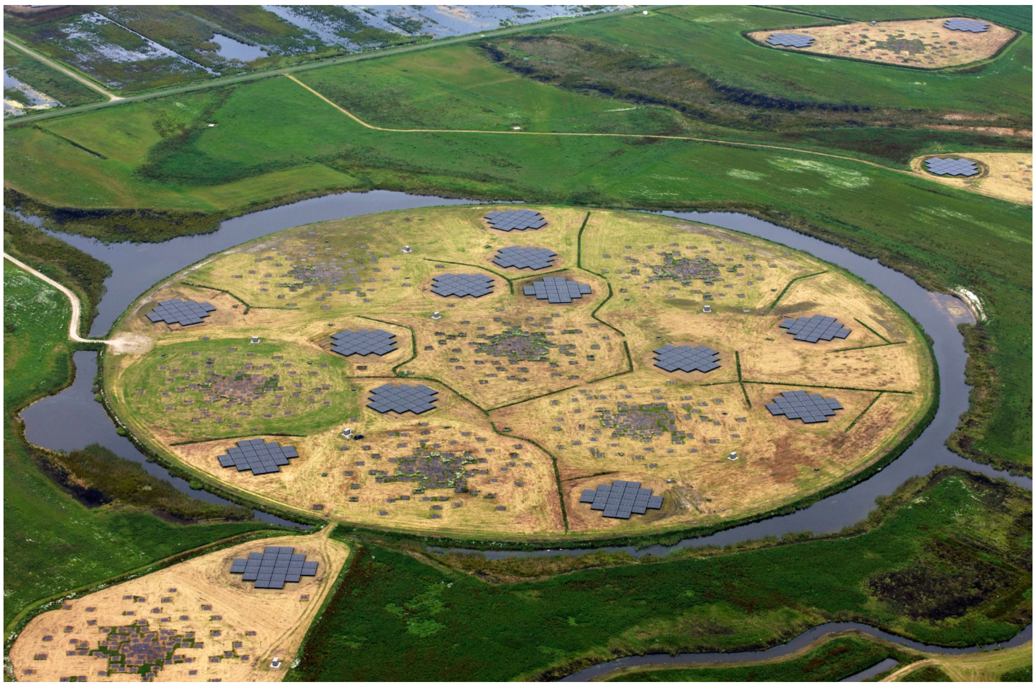


Figure 1.1.1: The LOFAR Superterp. The central six LOFAR stations visible with a few of the other stations in the LOFAR core visible in the foreground and background. Visible are high band antennas (HBA) in the form of hexagonal tile arrays and low band antennas (LBA) in the form of the 'haphazardly' arranged rectangles between each pair of hexagonal arrays. [11]

Comprised of 52 stations scattered across the Netherlands and other parts of Europe, LOFAR is capable of producing an impressive volume of data, nearly 35 TB per hour [11]. Despite these large figures, the primary practical consideration is the number of stations, as it dictates the number of data streams or correlated baselines that require inspection for data integrity.
Each baseline is represented by an inspection plot. These plots offer a human-readable visualization of the data, enabling anomalies to be easily identified. As such, various pre-existing visual techniques can be adapted for this purpose, as demonstrated by prior work on this issue by Mesarick et al. [18]. Their research utilizes self-supervised learning to develop a tool that aids manual inspection. Initially developed with synthetic data from the HERA simulator [19], the tool and associated model have since been adapted for specific use with LOFAR.

1.2 Thesis Outline

Given a large sample of LOFAR data in the form of down-sampled spectral data with general annotations per observation, this project aims to address several key issues. First the identification of anomalous behaviour and its categorization as separate from 'normal' data. The aim is to make a system that is able to learn a semantically salient representation of the data using self-supervised techniques and distribute this in latent space such that it enables further use for both categorization of already seen phenomena as well as out of distribution detection to spot novel sample instances when and as they appear.

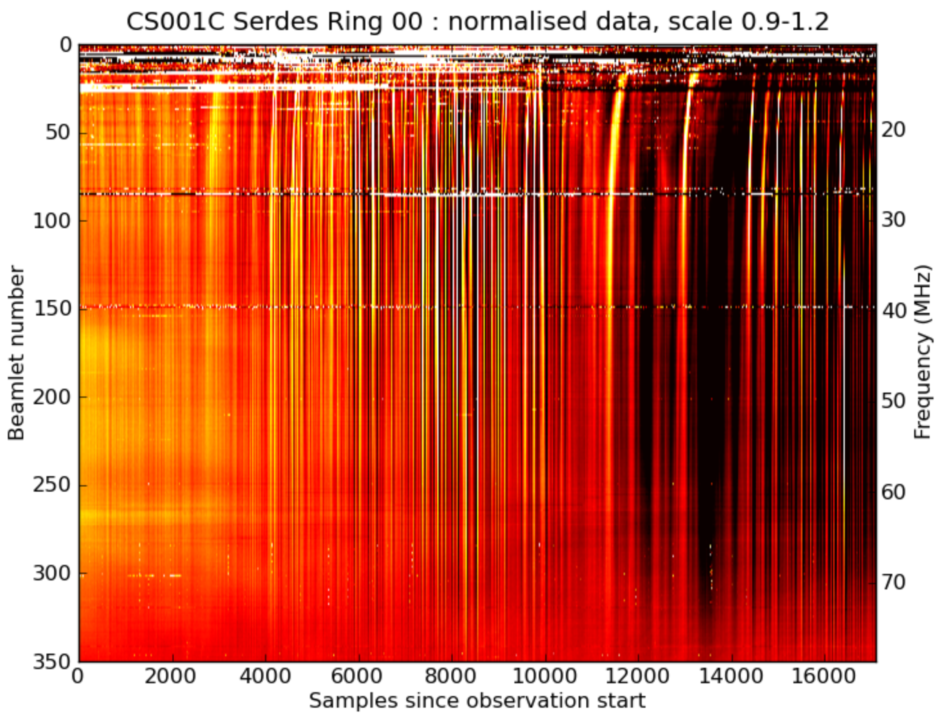


Figure 1.1.2: Example inspection plot showing strong scintillation. Closely spaced lines running from wider at the top (lower frequency) to narrower bottom (higher frequency) show this scintillation occurring. The additional horizontal frequency-invariant lines also shown in the plot are a separate but similarly anomalous/undesirable Radio Frequency Interference (RFI) pattern. Both of these phenomena show external sources of anomalies. This can be contrasted against system errors which would also manifest as anomalous data.

Primarily the research question becomes: **What is the most effective representation learning method to use on spectral LOFAR data to generate embeddings which are relevant for downstream tasks? Secondly; how effective are evaluations based on small validation sets with preliminary labels attached? How does this compare to unsupervised evaluation?**

In the course of this project, we propose a novel cascading convolutional auto-encoder using LOFAR specific context cues and a design geared towards learning a semantically relevant representation of spectral time-frequency data. This is then tested against four baseline methods including an adaptation of the Variational Auto-encoder method proposed in previous research on this topic [18]. The other methods include two further auto-encoder based methods and Independent Component Analysis; a well established linear transform whose primary application is blind source separation [20].
All of these methods are trained in a semi-supervised manner on the abundantly available LOFAR data, then evaluated against three tasks which fit the needs of the LOFAR system specifically. Namely these are data inspection, unsupervised data modelling by proxy of clustering and supervised evaluation.
The thesis is structured in the following manner; further theoretical background on the relevant concepts in radio astronomy and machine learning are given in Chapter 2. Following this, the creation of a labelled dataset and the setup experimental setup as well as model parameters are given in Chapter 3. Results are described in Chapter 4. Finally the results are discussed and used to answer the earlier

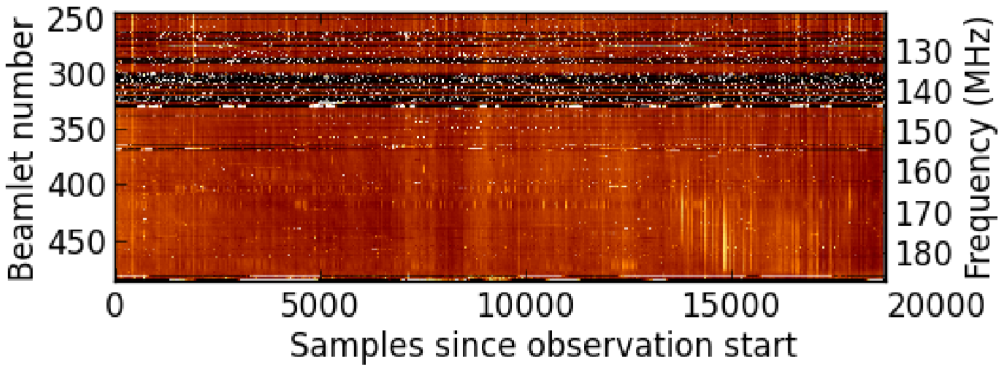


Figure 1.1.3: Example of 'clean' data. Note the persistent presence of horizontal lines in the data representing radio frequency interference (RFI). Such interference is very prevalent in all observations and there is very little possibility of obtaining data without such phenomena.

posed research question in Chapter 5.

Chapter 2

Theoretical Background

This chapter covers the essential background required to understand this thesis work. The segment covering LOFAR is constrained to details of the data collection and processing pipeline which are relevant to this work, with only minor points of interest straying outside of that bound. Following this there is an in-depth overview of the machine learning building blocks which play a part in this work as well as connections drawn back to past applications of machine learning on spectral data. This includes a specific focus on computer vision and representation learning methods as well as a few notes on the sub-field of anomaly detection.

2.1 Aperture Synthesis Radio Astronomy

Radio astronomy primarily focuses on the exploration of celestial entities within electromagnetic (EM) spectrum segments beyond the optical telescopes' purview. This field primarily deals with regions of the EM spectrum characterized by wavelengths larger than those of light, extending up to roughly 1 m. Additionally, there are applications exploiting detectors designed for very high frequencies, such as those encountered with gamma radiation.
A critical aspect of radio astronomy is the ”radio window,” a band within the EM spectrum where Earth-bound observations are possible. This window spans from about 1 mm to 30 m in wavelength [12]. Below this range, the atmosphere reflects waves, making observations challenging due to potential absorption of waves by certain atmospheric constituents above this band.
The methodologies of radio astronomy can be generally categorized into filled aperture and aperture synthesis approaches. The former involves singular large dishes, exemplified by now-inoperative instruments like the Arecibo telescope. Conversely, aperture synthesis employs an array of smaller receivers, combining their signals to produce a comprehensive image of the observation target. This technique is embodied by installations like the VLA in New Mexico [21], the Westerbork Synthesis Radio Telescope [22], and LOFAR [11], which is the primary focus of this thesis.
These instruments, particularly LOFAR, operate in close connection with the limitations imposed by the atmospheric radio window. For instance, LOFAR operates at frequencies as low as 20 MHz, nearing the 10 MHz boundary where radio waves are more likely to reflect off than permeate the ionosphere. Furthermore, at frequencies up to about 100 MHz, certain anomalies related to reflection and scintillation can be detected, as the reflection characteristics of these waves are heavily dependent on the local conditions of the Earth's ionosphere.
Aperture synthesis is a technique used in radio astronomy to create high-resolution images of astronomical objects beyond what is possible with a single telescope. Rather than a single receiver, it involves combining the signals from multiple radio telescopes all observing the same target to create

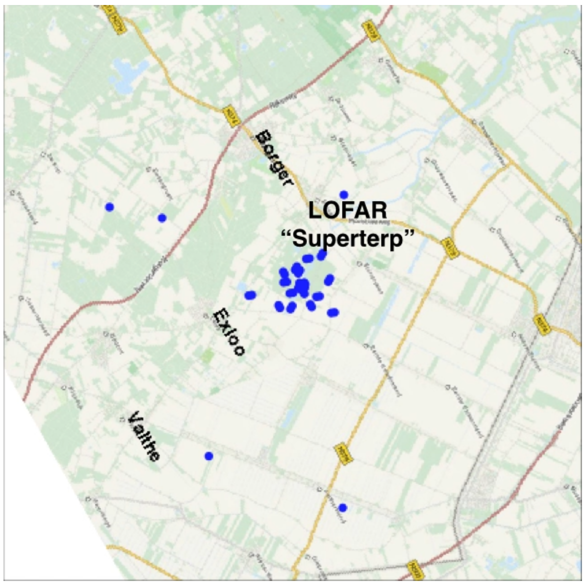
a synthesized aperture which is equivalent to a single, much larger telescope. This allows for much greater angular resolution when observing distant sources meaning better overall spatial resolution of the targets. Underpinning this is the process of combining these disparate signals into one coherent observation, this process is called interferometry. Interferometry is the process of taking signals of a common origin recorded at different locations and combining them with a time delay in order to to remove noise and obtain information about the properties of an object given its radio interference patterns [23]. The signals first have to be carefully aligned in order to compensate for the time delays from the signal hitting different antennae at different times. The basic interferometric correlation is given by the following formula:

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x_x(t) x_y^*(t-\tau) dt \tag{2.1.1}$$

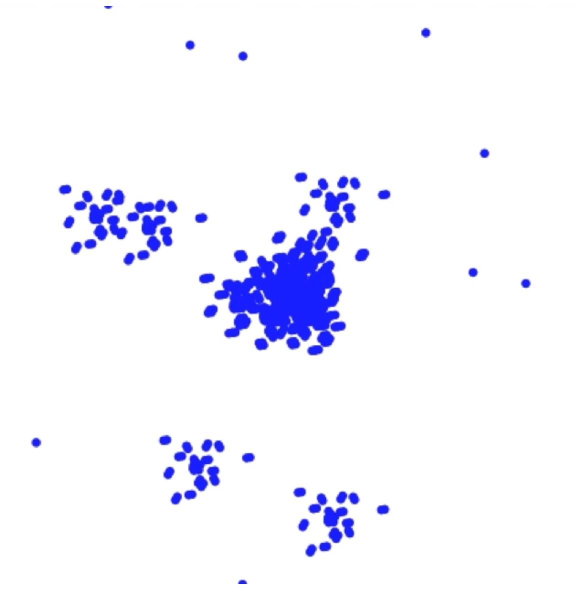
$R_{xy}(\tau)$: This is the cross-correlation function. It measures the similarity between two signals $x_x(t)$ and $x_y(t)$ as a function of a time-lag applied to one of them, which is τ in this case. τ: This represents the time delay between the signals. $x_x(t)$ and $x_y(t)$: These represent the signals received at antennas x and y respectively. $x_y^*(t-\tau)$: This is the complex conjugate of $x_y(t-\tau)$. The complex conjugate is used here because the signals are generally complex in nature (having real and imaginary parts), especially in the context of radio interferometry. The integral from $-\infty$ to ∞: This signifies that we're considering the signals over all time. In practice this is a little different because the signals are not infinite. However the assumptions made by this hold so long as $t >>> 1/\omega$ where $\omega = 2\pi v$ is the angular frequency of the observation.
LOFAR is an array of receivers where each receiver is again composed of smaller receivers. For example, the stations as can be seen in Figure 1.1.1 show the HBA units are themselves composed of individual tiles (which are underneath split further into dipole antennas embedded in a rigid structure). This means that the phase adjustment step is performed once already at the station level before being sent on to the central correlator where this is done again between pairs of stations before the correlation step. Of the core stations at least, each has two distinct HBA clusters which are treated as separate antennas/stations and are correlated as well into a baseline. This means that despite the fact that LOFAR only includes 52 stations, there are effectively a total of 76 HBA antennas in the system. To understand why this is relevant, let's have a look at how the number of baselines recorded grows with the number of antennas in the system. This is given by the following formula:

$$\frac{1}{2}(N^2 - N) \tag{2.1.2}$$

Where N refers to the number of stations in the system. This means that LOFAR with a total of 76 HBA stations produces about 2850 baselines in such an observation. The LBA antennas are not split as such and therefore only produce about 1326 correlated baselines. Another important element is the baseline length, meaning the distance between the pair of stations for which the correlation is being computed. This distance is measured in multiples of λ where λ is the wavelength at which the observation is being performed. This is because this distance, in the later computed Fourier space of the correlation determines the frequency of the interference patterns which are seen in the baseline plots. This parameter additionally relates back to equation 2.1.1 as the time delay τ is determined by this distance.
The image plane in aperture synthesis serves a similar function as the image sensor with pixels in an optical camera, once you account for the mathematical transformations required. As more data is collected from more stations (each pair providing a unique baseline), the final image becomes increasingly refined. In this context, the LOFAR telescope collects observations over several minutes



(a) LOFAR Core Stations - Map



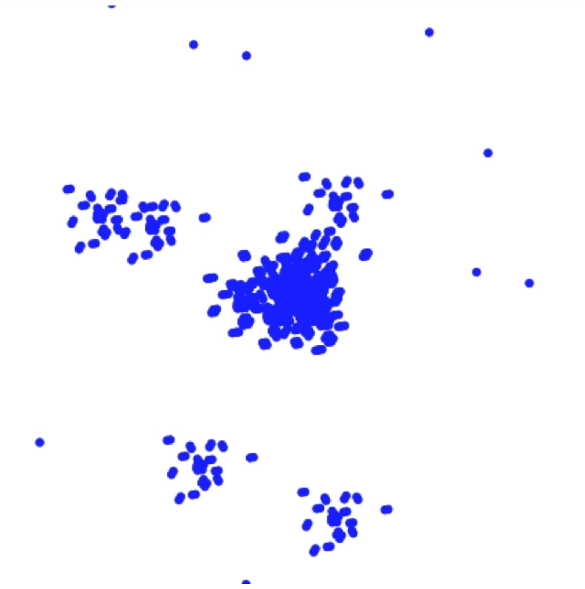
(b) LOFAR Core Stations - Fourier Plane

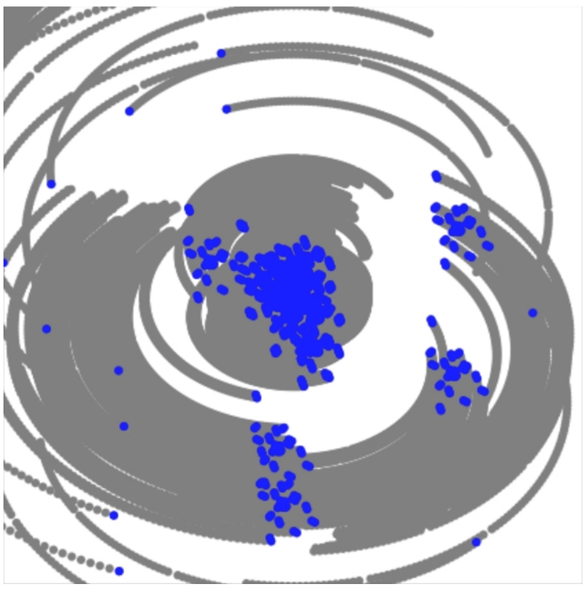
Figure 2.1.1: Images showing how LOFAR physical station locations are mapped to Fourier space coordinates. Images retrieved from LOFAR informational video. Retrieved from [24].

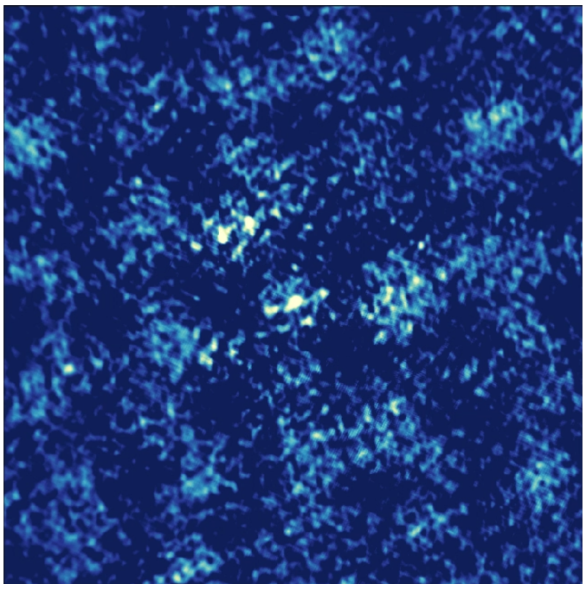
or even hours to adequately populate the image plane. The coordinates within this plane are calculated based on the target's right ascension and declination, and the position of the stations relative to each other forms the baseline distance. For this research, the correlated data from a pair of stations will be referred to as a 'baseline'. Figure 2.1.1 provides an example of the image plane from LOFAR's central stations. The total number of baselines is calculated relative to the total number of stations according to the formula in 2.1.2.

Now, imagine the image plane as an array that yields a higher resolution image as more of its elements are filled. By extending the observation period from a few seconds to a few hours, we can compare the completeness of the image plane with the quality and clarity of the resulting image. This process is illustrated in Figure 2.1.2. Therefore, the more observations that can be integrated into this array, the better the resolution and the greater the resilience against noise. However, this process does have a caveat: inaccurate or corrupted data can actually hinder the image reconstruction. So, it is crucial to remove such data to prevent it from introducing noise or confounding factors that could significantly impact the final image quality.

Finally, correlation or the main component of the interferometry as described in equation 2.1.1 is done per frequency bin that the array supports. The technical documentation puts this at a maximum resolution of 195 kHz per bin [11], meaning that a typical observation made in the High Band Antenna (HBA) range of 110-240 MHz will have around 200-250 frequency bins. This is slightly fewer for the Low Band Antennas (LBA) where the frequency ranges it is able to record range between 20 MHz and 90 MHz. This forms a discontinuity in the designed observation range of 30-240 MHz. For this reason, later when referring to the data from the array the reader should assume that the frequency band will be specified. If this remains un-specified, assume that the observation is made in the HBA range. The correlation is also done per polarization. Both sets of antennas (HBA and LBA) are configured as cross shapes, meaning that they are able to record radio waves in both an X and Y polarization configuration. If this is the case, the resulting data will include the pairwise correlated polarizations which is to say (XX, YY, XY, YX).


(a) Initial Fourier plane state - no movement


(b) Fourier plane state after 8.5 hours observation


(c) Initial snapshot sky image reconstruction at first point of observation

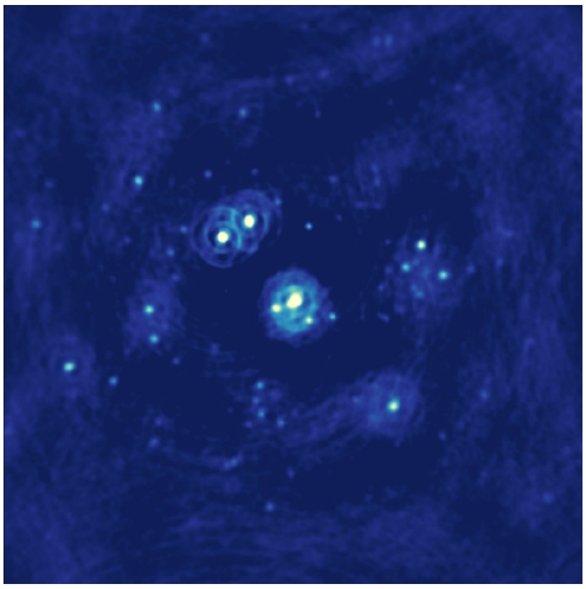

(d) Refined reconstruction after 8.5 hours of observation

Figure 2.1.2: Figures showing how filling out the Fourier plane with more data points provides more data to de-noise observation and achieve better resolution. Images retrieved from LOFAR informational video [24].

2.2 Machine Learning

In broad terms, machine learning is a sub-set of techniques in AI used to create models capable of certain desirable tasks such as image classification [25] or segmentation [26] and which improve with the amount of data given to them to train on. There are, of course, several sub-segments of machine learning as well. The key segments being *supervised learning*, *unsupervised learning* and *self-supervised learning* as well as reinforcement learning. This thesis concerns mostly self-supervised and unsupervised learning, however it is useful to contextualize them against supervised learning in order to gain a better understanding of the problem space.

2.2.1 Supervised Learning

Supervised learning, an integral pillar of machine learning, operates on the principle of learning an approximating function f from given input-output pairs. This process hinges on a training dataset comprised of input features X and corresponding target outputs Y, which are presumed to be generated by the function f. The task of supervised learning, therefore, is to construct an approximation of this function, denoted f', that can accurately map inputs to their respective outputs.
Central to the supervised learning paradigm is the iterative process of prediction and correction. Given an input $x \in X$, the model generates a prediction $y' = f'(x)$. The discrepancy between this predicted output y' and the actual output $y \in Y$ is quantified by an error metric or loss function g. This error $e = g(y, y')$ is a measure of the deviation of the model's prediction from the true output.
The learning process iteratively uses this error information to adjust the parameters of f', with the aim of minimizing the overall loss. Ultimately, a well-trained model, through these cycles of prediction and error-based correction, can generalize well to unseen data.
Supervised learning finds extensive applications in tasks such as image classification [25] and is foundational to understanding other learning paradigms, such as self-supervised and weakly supervised learning. A more nuanced introduction of error functions and optimization techniques, as well as the extension of these principles to related learning methods, will be provided later in this section.

2.2.2 Unsupervised Learning

Unsupervised learning, another essential paradigm in machine learning, thrives in the exploration of unlabeled data and its inherent patterns, with no explicit expected output to target. Unsupervised learning predominantly covers clustering and other distribution or partition learning methods. Some examples of clustering methods include k-Means clustering or fuzzy c-means being a hard and fuzzy partitioning methods respectively. Distribution modelling includes such methods as Dirichlet process clustering and Gaussian Mixture Models. Different density-based methods include DBSCAN and OPTICS. To round out the gauntlet of clustering methods, hierarchical methods should be mentioned, such as agglomerative clustering. There is therefore a myriad of methods available for unsupervised data analysis. This thesis focuses primarily on k-Means clustering and Gaussian Mixtures, more on those particular methods in Section 2.6.
In the absence of target data however, the matter of performance metrics in the unsupervised learning case comes into question. How, for instance, does one ascertain the appropriateness of a derived clustering solution? Several evaluation methods have been proposed for this, falling under the umbrella term of clustering validity indices. Some basic examples include the elbow method [27], measuring the frequency of co-occurence of two distinct points in the same cluster [28] or otherwise defined evaluation metrics such as the silhouette score [29]. Further exploration of these methods of evaluation is

made in Section 2.7.

2.2.3 Self-Supervised Learning

Self-supervised learning (SSL) represents a paradigm within the broader context of unsupervised learning, where the objective is to discover insightful and compact representations of data. The underlying principle of SSL is to exploit the intrinsic structure of the data to derive supervision signals, thus negating the need for externally annotated labels.
The central tenet of SSL involves training a model to perform a task that is defined exclusively using the input data itself. Formally, given a function f parameterized by weights θ, the goal is to find an approximation x' such that a specific error function g attains a minimum value, $\min_\theta g(x, x')$. The task itself requires the identification of a meaningful yet constrained representation of the data, either through undercomplete or overcomplete mappings with associated sparsity or other regularization constraints.
The superficial resemblance between supervised learning and SSL can mislead; the latter distinguishes itself through the innovative utilization of data. Unlike traditional supervised learning that requires separate input X and target Y, SSL leverages the same data X as both input and target. This offers the dual advantages of maximizing the utilization of available data and reducing reliance on expensive human annotations.
A crucial aspect of SSL involves the design of pre-conditioning tasks or pretext tasks, which enable the model to generalize better. These tasks are often devised to reflect essential properties of the data or the specific downstream task at hand. Examples include:

- **Masking Strategies:** Portions of the data are randomly masked or corrupted, and the network is trained to reconstruct the original data, thereby forcing it to learn salient features.
- **Contrastive Learning:** Utilizing pairs of similar and dissimilar data points to make the learned representations discriminative.
- **Data Augmentation:** Applying systematic transformations (e.g., rotations, scaling) to the data and requiring the model to predict the transformations or to recognize the unaltered data.

These pretext tasks, by virtue of their intrinsic nature, drive the model to learn universal features that are expected to be beneficial for various downstream tasks. This emphasizes the essence of self-supervised learning: learning valuable representations from data itself, without explicit external supervision, but with the guidance of carefully designed tasks that align with the underlying data's inherent structure.

2.3 Dimensionality Reduction and Representation Learning

Dimensionality reduction techniques play an indispensable role in machine learning and data analysis, as they provide a mechanism to transform high-dimensional data into a lower-dimensional format. This transformation preserves the intrinsic structure of the original data, making it easier to handle and interpret. Dimensionality reduction methods are broadly categorized into linear and non-linear types.
Linear methods, such as Principal Component Analysis (PCA) and Independent Component Analysis (ICA), involve linear projections of the high-dimensional data into a reduced-dimensional space.

These methods function on the premise of capturing the largest variance in the dataset or learning components that can reconstruct the data through linear combinations.
On the other hand, non-linear methods, such as t-Stochastic Neighbour Embedding (t-SNE), engage more complex approaches like manifold learning to achieve a similar goal. t-SNE is a widely used technique for data visualization due to its ability to represent high-dimensional data in two or three dimensions while preserving its overall structure.
An integral assumption for these techniques is the consistency of features across the dataset. This means that each element of the data vectors must represent the same feature in the data. If this consistency is not maintained, the efficacy of these dimensionality reduction methods may be compromised.
These techniques often serve as precursors to more sophisticated neural network-based approaches such as auto-encoders. Auto-encoders are particularly useful for representation learning, which involves transforming the data into a lower-dimensional space where the new representation retains some semantic properties of the input.
In the following sections, we delve deeper into PCA, ICA and t-SNE. Auto-encoders are revisited later in Section 2.5.1.

2.3.1 Principal Component Analysis

Principal Component Analysis or PCA for short is a widely applied method of dimensionality reduction [30]. It finds linear combinations of variables which define the axes of greatest variance in the data. Mapped and sorted as a per-feature histogram where each feature is ranked by its amount of explained variance (how important it is to describing the overall shape of the data) we generally find that there is a large tail of features which add little to the overall 'explanation' of the shape of the data. This allows us to start cutting from this tail of the distribution features which are less relevant. While this is a lossy operation, the assumption is that the most informative features are still kept, leaving the structure of the data largely the same. PCA is calculated as follows; first, given data matrix X we compute its covariance matrix

$$C = \frac{1}{(n-1)} X'TX' \tag{2.3.1}$$

Where X' is the normalized matrix X such that $X' = X - \hat{X}$ where $\hat{X}$ is the mean of matrix X. T in this case is the transpose of matrix X'. We then apply to this covariance matrix *singular value decomposition* (SVD). While the original formulation of PCA uses eigenvalue decomposition, we apply singular value decomposition instead as it yields the same results, but has some favourable computational properties. Our singular values are of the same magnitude as the eigenvectors indicating the variance explained by each feature in the projected space. So then, given our singular value decomposition

$$C = U\Sigma V^T \tag{2.3.2}$$

We then use these components to create a more suitable projection of the data sorted by its eigenvalues indicating the magnitude of variance in a particular dimension of this projection.
This method is not scale and transformation invariant, meaning that it only works well when each feature in a data vector means the same thing between two consecutive data samples. Although this is the case the method has been successfully applied to images [31]. Though, even with the images it is necessary to ensure, beforehand that the features under analysis appear at the same scale and roughly in the same positions from image to image. They key takeaway here is; the PCA transform results in a de-correlated dataset, which is a pre-condition for the next method described; Independent Component Analysis.

2.3.2 Independent Component Analysis

A linear method of dimensionality reduction which is rather more suited to signal processing is Independent Component Analysis (ICA) [32]. To describe why ICA is appropriate for signal processing, we should first explore its purpose. Given m mixed signals ICA will attempt to find n source signals where $n < m$ and at most one of the signals n is Gaussian in nature. This is called *blind source separation* and provides us with a very powerful means of de-mixing any number of signals with an unknown prior mixing process [20].

In practice this method can be used on images as well to a similar effect to find repeating underlying patterns in visual data [33]. The original image should then be able to be reconstructed as a linear combination of these underlying patterns.

Unlike PCA's assumption of co-linearity, ICA assumes that the disparate portions of a given signal are both non-Gaussian and statistically independent. It uses this property to learn both component filters or signals and a demixing matrix which, given the filters would reform the original signal again. This can be considered a type of representation learning then as it is now possible to decompose a signal or even an image into a series of signals. Given the assumption of non-Gaussian component signals, ICA can furthermore be used for spotting and classification of persistent noise patterns. This has been applied to great effect, for example in EEG research [34]. However in cases where the assumption of non-gaussian components is somewhat weaker, methods are available to distinguish between useful sources and sources which themselves may be some weakly learned combination of various Gaussian noise components [35].

2.3.3 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a high-dimensional data visualization algorithm [36]. t-SNE is particularly adept at preserving local structure within the data while maintaining relative global arrangements. Unlike linear techniques such as PCA, t-SNE employs a probabilistic approach to model similarity between data points, making it better suited for embedding non-linear data manifolds. t-SNE accomplishes this by first measuring similarities in the high-dimensional space and then minimizing the divergence between these measures and a t-distribution fit in the lower-dimensional space.

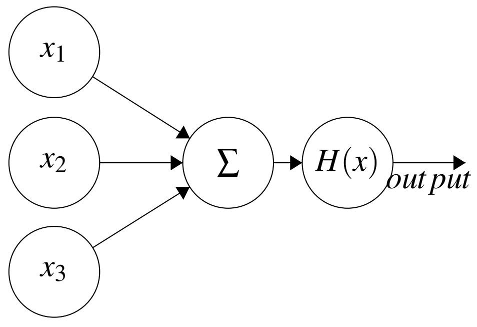
The formulation of t-SNE involves a series of steps. First, it transforms the Euclidean distances between high-dimensional data points into conditional probabilities, representing the likelihood that one point would pick another as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at the point. For a pair of points x_i and x_j in the high-dimensional space, the conditional probability is given by

$$p_{i|j} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{2.3.3}$$

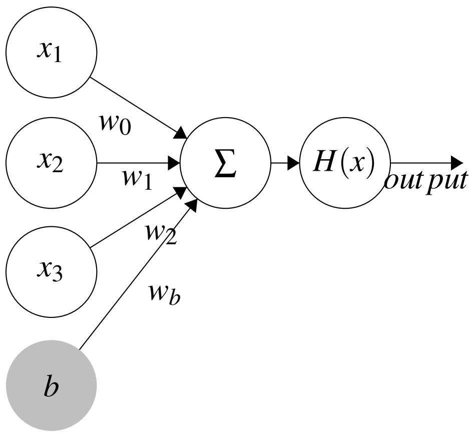
where σ_i is the variance of the Gaussian that is centered on x_i.

The similarities in the low-dimensional space are calculated similarly, but with a key difference - instead of using a Gaussian distribution to compute the conditional probabilities, it uses a Student's t-distribution with one degree of freedom (a Cauchy distribution) to do so. The joint probabilities in the low-dimensional space for points y_i and y_j are then given by

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k\neq l}(1 + ||y_k - y_l||^2)^{-1}} \tag{2.3.4}$$



(a) McCulloch-Pitts Neuron



(b) Rosenblatt Perceptron

Figure 2.4.1: A side by side graphical comparison of the formulation differences between the McCulloch-Pitts neuron (Left) and the Rosenblatt Perceptron (Right). The McCulloch-Pitts neuron takes N inputs which are summed and passed through an activation function $H(a)$ to produce an output of either 0 or 1. The Rosenblatt Perceptron takes N weighted inputs and one bias term b which are summed and passed through a step activation function to produce an output of 0 or 1.

The t-SNE algorithm then minimizes the divergence between the high-dimensional and low-dimensional distributions, typically using the Kullback-Leibler divergence (see Section 2.4.4):

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{2.3.5}$$

Here, P and Q represent the joint probability distributions in the high-dimensional and low-dimensional space, respectively. This cost function is minimized using a gradient descent method.

2.4 Artificial Neural Networks

Artificial Neural Networks (ANNs) are among the more powerful techniques which can be used in machine learning. Inspired by the human brain, these networks are both versatile in their wide range of potential application domains as well as powerful in being able to act as approximators for arbitrary functions (given the correct conditions). ANNs have been applied to regression tasks [37], to categorical classification [38] and control tasks [39]. Before we get too far ahead of ourselves however, it is salient to look into the basic building blocks of these ANNs to see how they are able to act as such powerful function approximators.
This section covers the concept from the basic building blocks, through the introduced non-linearities and why they are important to both the ANNs ability to approximate arbitrary functions and its optimization process. Following this we look at some common loss functions as well as the learning and optimization process, ending up finally in the realm of more modern ANNs with the Convolutional Neural Network (CNN) architecture and its applications to image-based tasks.

2.4.1 Perceptron

The conceptual basis of all artificial neural networks is its most basic building block; the artificial neuron. The first iteration of this concept was devised in a 1943 paper by McCulloch and Pitts forming the basis for a simplified mathematical formulation of a biological neuron [40]. This neuron takes and sums a set of $\{x_1..x_n\}$ inputs to produce an activation as shown in equation 2.4.1. Then the activation level is compared to a threshold value θ based on which the activation is determined to be either 0 or 1 as shown in equation 2.4.2. This mirrors the 'all or nothing' response found in biological neurons.

$$\sum_{i=1}^{n} x_i = a \tag{2.4.1}$$

$$H(a) = \begin{cases} 0 & \text{if } a < \theta \\ 1 & \text{if } a \geq \theta \end{cases} \tag{2.4.2}$$

While the McCulloch-Pitts concept of the artificial neuron is a sensible mathematical model by itself, the Rosenblatt Perceptron [41], a formulation which adapted this with certain additions is what allows this basic building block to go from a simple response mechanism to a workable method for iterative learning. Namely, the added parameters are weights on each of the inputs to a neuron unit as well as an additional bias term. The input summation function is then represented in equation 2.4.3. As can clearly be seen, if the input space is reduced to a single item, the result is a standard linear function $y = wx + b$. Next, the activation is described by equation 2.4.4. This is a simple Heaviside step function with the main difference to the threshold function being that it is centered around a 0 meaning that any positive non-zero activation causes the neuron to fire.

$$\sum_{i=1}^{n} w_i x_i + b = a \tag{2.4.3}$$

$$f(a) = \begin{cases} 0 & \text{if } a \leq 0 \\ 1 & \text{if } a > 0 \end{cases} \tag{2.4.4}$$

To distinguish between the two types of proposed neuron, refer to the visual representation of each formulation in Figure 2.4.1.
Given the weight vector at state t as $\vec{w}(t)$, a learning rate of α and an input/target pair of $(\vec{x}_j, y_j)$ with y'_j being the output of the perceptron at step t given input x, the update in weights is calculated using the formula described in equation 2.4.5. This is a weighted update step and this learning process is called the Hebbian rule or more generally, Hebbian learning is being applied to the perceptron.

$$w_i(t+1) = w_i(t) + \alpha * (y_j - y'_j)x_{ij} \tag{2.4.5}$$

2.4.2 Multi Layer Perceptron

The drawback of a single perceptron is that it is only able to learn linearly separable functions. This means function such as AND, OR are possible to learn, however XOR is not linearly separable, meaning a different approach is needed. So while a single perceptron is a powerful tool in itself, it would be more useful to be able to learn approximations of non-linear functions as well by stacking multiple layers of perceptrons. This idea forms the basis of the Multi Layer Perceptron (MLP). One remaining issue is that simply stacking several perceptrons does not achieve the desired effect of being able to approximate non-linear functions, a further element is required. The last piece of the puzzle are non-linear activation functions. Without them the MLP may be able to approximate very complex linear functions and combinations thereof, but non-linear functions would still be out of reach.
Figure 2.4.2 shows a simple MLP with some inputs, one hidden layer worth of units and two outputs. Conceptually this forms a directed acyclic graph where information flows from the input to the output nodes. Representing the weights between the units as matrices is convenient because it allows

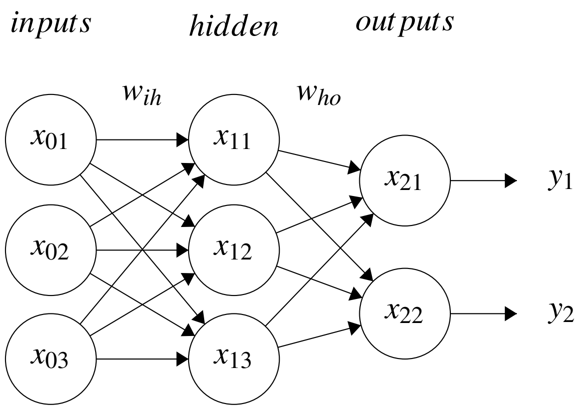


Figure 2.4.2: Diagram of a Multilayer Perceptron with one hidden layer. w_{ih} and w_{ho} show the weights of the layer connections, which it is convenient to imagine as matrices.

the understanding of our 'forward pass' of this graph to be expressed as a simple series of matrix multiplications. The activations $A_h(x)$ of the hidden layer can be expressed by the following formula:

$$A(x) = g(x^T w_{ih} + b) \tag{2.4.6}$$

Where g represents the non-linear activation function, w_{ih} is the matrix of weights between the input and the hidden layer, x is the matrix of inputs and b is the bias term. The same process is then repeated with w_{ho} between the hidden layer activations and the outputs. Having mentioned the 'forward pass' or, the 'backward pass' or learning step of this procedure is deferred to later in Section 2.4.5.

2.4.3 Activation Functions

As mentioned above, the key to having a neural network composed of artificial neurons be able to solve non-trivial tasks is an appropriate choice of activation function. A good activation function should meet two criteria, first it must be non-linear and second its derivative should ideally never be zero. This latter requirement combats the problem in optimization of vanishing gradients [42].

ReLU - The Rectified Linear Unit or ReLU proposes a very simple non-linear function to be introduced in the activation stage of a neural network. It introduces a non-linear function in the form of a discontinuous linear function and a nulling element. This function is linear when the activation is positive and zero everywhere else.

Leaky ReLU - The standard rectified linear unit can take us a very long way , however it does not satisfy our second condition which is that the derivative of the function should be non-zero at every point in the function space. The leaky ReLU solves this issue by modifying the function in the interval $[-\infty, 0]$ so that rather than being set to zero, the function is again linear but with a much smaller slope, usually 0.01. A plot of the Leaky ReLU activation and its derivative can be found in Figure 2.4.3 where the slope in the aforementioned interval is exaggerated to 0.2 to clearly differentiate it from the base ReLU.

ELU - Another activation function which attempts to solve the ReLU problem of weak gradient transition around 0 as well as below 0. This function addresses this by introducing an additional

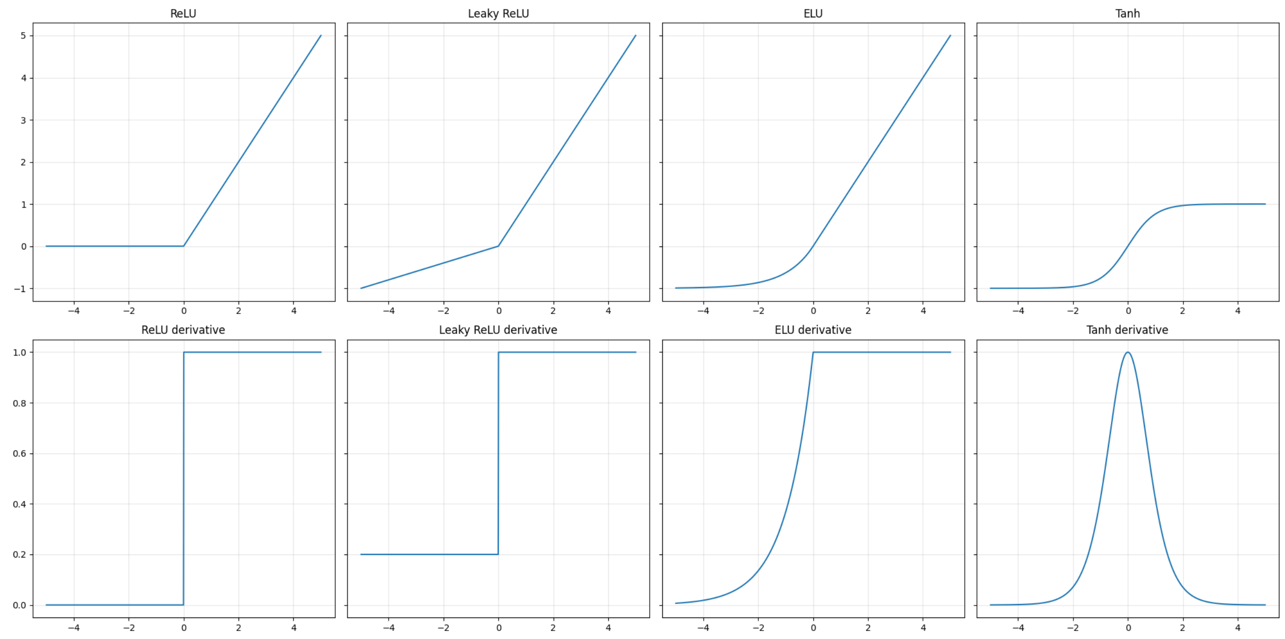


Figure 2.4.3: Comparison of 4 activation functions (top) and their derivatives (bottom).

parameter; an alpha value which determines the negative value which the function returns to asymptotically, this mostly affects the shape and definition of the gradient below 0 and remains the same as the ReLU and Leaky ReLU above 0.

Sigmoid - Sigmoid activation must be mentioned here as it is adjacent to the step activation function used in the perceptron and is itself used in the basic multi-layer perceptron model. It is also closely related to the next function on the list; the tanh activation function. The sigmoid activation function is bound between 0 and 1 on the y-axis with weak to zero gradients far from 0 on the x-axis. This function is useful for ensuring that the output of a layer is bounded in a predictable range of values, however far from 0 on the input backpropagation may struggle with weak gradients meaning it is possible to get dead neurons.

Tanh - Similar in shape and derivative to the Sigmoid function, the Tanh activation function is instead bounded between -1 and 1 on the y-axis. This addresses one of the shortcomings of the Sigmoid function namely that it is asymptotic to 0 at highly negative values. This may result in entire sections of network which remain unused. Primarily we are interested in this function because of its asymptotes at -1 and 1 producing something akin to normalized output.

2.4.4 Loss Functions

Given that training a neural network is a function minimization problem, there has to be a function that facilitates this optimization. The class of functions which compute this error between expected and actual output are called loss functions. Because they inform our optimization, much like activation functions, these must be differentiable in order to be applicable.
Loss functions are typically applied to the output of a network, however this is not a requirement. Certain loss functions may also be applied to intermediate states in a network in order to produce some kind of regularization effect. We will explore this further in parts of Section 2.5.1.

Mean Absolute Error Loss MAE Loss also known as L1 Loss is conceptually the most basic error function available for use when comparing two continuous values. For an n length vector of values $\hat{Y}$ and target values Y the error is calculated as specified in formula 2.4.7

$$MAE = \frac{1}{n}\sum_{i=0}^{n} |\hat{Y}_i - Y_i| \tag{2.4.7}$$

This simply calculates the absolute deviation between the obtained value and the expected value for all values in the result/target vectors. To simplify the contextualization of the next presented loss function, one can imagine that the deviation calculation portion of this L1 loss function is raised to the first power, so $|(\hat{Y}_i - Y_i)^1|$. This formulation will become clear with the introduction of the next loss function.

Mean Squared Error also known as L2 Loss is very similar to Mean Absolute Error, but instead the principal deviation calculation is raised to the second power. This is where the formulation in the L1 loss section comes in handy as the reader should see parallels immediately between this calculation and the last. See formula 2.4.8 for details on how it is calculated.

$$MSE = \frac{1}{n}\sum_{i=0}^{n} (\hat{Y}_i - Y_i)^2 \tag{2.4.8}$$

As can be seen, all but the central deviation computation here remains the same and even there only the power raised to is changed. This results in an error calculation that amplifies the error for values which more heavily deviate from the target. This is useful as it allows for a very quick convergence in general to a 'good enough', however MSE has a tendency to over-represent outliers in data and miss very small changes [43]

Wasserstein Distance Metric or "earth mover's distance" is a measure of similarity between two distributions [44]. In the one-dimensional case, the distributions are compared using their cumulative distribution functions. Given univariate distributions P and Q with corresponding density functions $P(x)$ and $Q(x)$ for $x \in \mathbb{R}^1$, the Wasserstein distance can be calculated as follows:

$$R(P,Q) = \int_{-\infty}^{\infty} |P(x) - Q(x)| dx \tag{2.4.9}$$

For a fixed sample space of N samples, where each point samples the cumulative distribution density at that point, the Wasserstein distance can be approximated as:

$$R(P,Q) = \frac{1}{N}\sum_{n=0}^{N} |P(n) - Q(n)| \tag{2.4.10}$$

The complexity of calculating the metric grows exponentially with the number of dimensions in which the metric is calculated. However, an estimation trick called the Sliced Wasserstein Distance (SWD) [45] can be used to mitigate this issue. In SWD, multivariate distributions are projected onto K univariate vectors sampled from a unit hypersphere in $\mathbb{R}^d$, where d is the dimensionality of the compared point clouds. These projected samples are then used to evaluate the distributions in their univariate decompositions, resulting in the SWD formula:

$$SWD(P,Q) = \sum_{k=0}^{K} \frac{1}{N}\sum_{n=0}^{N} |P_k(n) - Q_k(n)| \tag{2.4.11}$$

Here, $P_k(n)$ represents distribution P sampled along random vector k in d dimensions, at the nth point in the cumulative density function. Similarly, $Q_k(n)$ represents the same mapping for distribution Q. It is important to note that SWD is an estimation of the true Wasserstein distance, and there are considerations regarding its robustness and stability, which are addressed in the following papers [46, 47].

KL Divergence is another method of comparing two distributions using a measure called *relative entropy* [48]. However, unlike the commutative nature of the Wasserstein distance metric ($SWD(P,Q) = SWD(Q,P)$), the Kullback-Leibler (KL) divergence is not commutative. The KL divergence from distribution P to distribution Q, denoted as $D_{KL}(P||Q)$, is a divergence and is calculated as follows:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{2.4.12}$$

In the context of the KL divergence, $P(x)$ and $Q(x)$ are used to denote the density functions of distributions P and Q respectively.
In general the difference between the KL-Divergence and the earlier presented Wasserstein metric is that KL-Divergence is somewhat more sensitive to very small changes between distributions. This effectively means that in the region of diminishing returns on optimization, the KL-Divergence might give better performance as a metric.

Reconstruction ICA is a loss statistic which, in contrast to all the previously presented similarity measures, rather enforces a sparsity constraint on a set of parameters. While it takes the name of the Independent Component Analysis method presented earlier in Section 2.3.2, it is a less stringent sparsity constraint [49]. Whereas ICA enforces a hard orthonormality constraint on the solutions it is able to find, RICA foregoes that in favor of a soft reconstruction loss. ICA is unable to deal with an overcomplete representation, for this RICA instead introduces a soft constraint instead of the hard orthonormality constraint imposed by ICA. This is defined as follows:

$$\min_{W} \quad \lambda||Wx||_1 + \frac{1}{2}||W^T Wx - x||_2^2 \tag{2.4.13}$$

Where W are the RICA weights and x is the input to the RICA layer. The second portion of this formula represents the reconstruction of the inputs and the first term is an 'activation penalty' on the intermediate result of the reconstruction. This is what promotes sparsity here. It should be noted that W and W^T are shared weights, meaning that it is the same set of weights each time.

2.4.5 Backpropagation and Optimization

While single layer networks can be updated using the perceptron update rule shown in Equation 2.4.5, once artificial neural networks become deeper and include more than one layer, the update function becomes somewhat more complicated. In order to best conceptually relay the problem, the task of updating a network can be split into three distinct steps. Firstly, the forward pass takes an input and calculates, based on current network parameters, the output. Based on the output a loss function is calculated. Given this information, one can then calculate the error as propagated to any of the network weights using back-propagation. Repeated application of the chain rule allows the partial derivative to be calculated at every point in the network. Finally the weight update step; we can see that the gradient at a node w_{ij} is given by

$$\Delta_{ij} = \frac{\partial e}{\partial w_{ij}} = \frac{\partial e}{\partial f}\frac{\partial f}{\partial w_{ij}} \tag{2.4.14}$$

Where $f(x)$ is the network function and $e(x)$ is the differentiable loss function used to compute the deviation from the desired score. This partial derivative can be calculated using repeated application of the chain rule. Given this, the weight update of node w_{ij} is then given by

$$w_{ij}^{new} = w_{ij}^{old} - \eta\Delta_{ij} \tag{2.4.15}$$

Where η is the scaling parameter or learning rate in this case which prevents the network from overfitting on single examples. Repeated application of these steps (forward, backward, optimization) is how one trains a network.
The intuitive way of thinking of this back-propagation, at least with regards to the most common operations used in neural networks (multiplication and addition) is that multiplication splits the gradient backwards based on the weight's total proportion of the multiplication operation it was in and addition simply propagates the gradient backwards 1:1 such that the gradient at every connected node backwards is the same as in the current node.
In gradient-based learning approaches one may face two types of problems. One wrought by improper choice of non-linearity in the activation layers or network depth; vanishing gradients, and its oppostite - exploding gradients. We have already addressed the problem of vanishing gradients somewhat with the non-linear activation functions, the other solution are residual connections (ie. skip connections that propagate larger gradients deeper into the network) [50]. Conversely, the exploding gradient problem occurs when the incremental network adjustments become too large. This can happen either due to a learning rate which is too high or due to lack of regularization. While the learning rate issue is addressed further in the methods, the basics of regularization are covered in the following section.

2.4.6 Regularization

Regularization is the name given to a collection of methods which attempt prevent reaching local minima during training. In most cases it is meant to prevent overfitting, which in the case of the autoencoder architecture is done by ensuring that the learned solution is undercomplete or sparse [51]. This means that this is an architecturally enforced constraint, however for something like exploding gradients which affects the optimization process directly different approaches must be taken. As mentioned in the above section on back-propagation and optimization, the exploding gradients problem may affect the quality of the optimization and in turn the quality of the solution. In this work, the preferred method of dealing with exploding gradients is simply ensuring that the learning rate is set properly. Batch normalization can be applied, while this is not directly a regularization technique it does have certain properties to that effect as well as providing other benefits in the course of training [52]. Batch normalization is described by the following formula:

$$\hat{x} = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}} \quad \text{and} \quad y = \gamma\hat{x}+\beta \tag{2.4.16}$$

The equation is split into two parts to represent the static portion and the learnable portion of a batch normalization layer. $\hat{x}$ is the normalized input batch x, however this is still scaled with a parameter γ and bias β is also applied. Both of these are learnable parameters, much in the same way that these terms apply to a regular layer in a network.

Another method applied in this research is RICA, however this has been described in some detail already in Section 2.4.4. Suffice to say here that it enforces some degree of sparsity on the network by means other than L1 regularization which has very good properties regarding the generalization potential of a network.

2.4.7 Convolutional Neural Networks

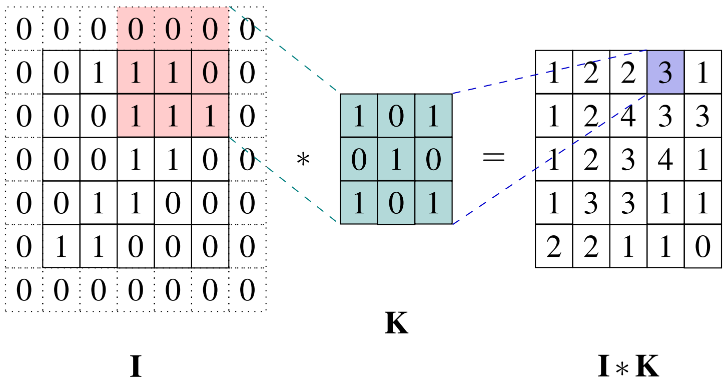


Figure 2.4.4: Example of how a convolutional kernel computes activation over a 2D image with one single channel. Note in this example the input is padded with 0s as it would be in a normal input scenario as well. This ensures that the original image size is preserved rather than cutting $k//2$ off each side of the image during the process where k is the size of the kernel.

Convolutional Neural Networks (CNNs) represent an advanced architecture of artificial neural networks that is often utilized in image processing and other types of spatial data analysis. A pivotal characteristic that differentiates CNNs from standard feedforward neural networks is the principle of weight sharing. This denotes that the same set of weights, referred to as a kernel or filter, is utilized to perform a convolution operation on different portions of the input data. This procedure is performed at regular intervals across the data, thereby providing the network with a powerful capacity to learn and recognize location invariant features.
In other words, CNNs are not restricted to recognizing patterns at specific locations within an image, as is the case with feedforward neural networks. Instead, the shared weights allow the networks to detect the same pattern regardless of its position in the input space. This weight sharing and spatial invariance constitute a fundamental characteristic of CNNs, allowing them to excel in tasks such as object detection within images, where the object of interest may appear at any location. Notably this flexibility does not extend to scale or rotation invariance, where CNNs do not deal with variation very well.
The filters within CNNs, which form the learnable parameters, are not predetermined as in certain specialized applications such as Sobel edge detection [53]. Instead, they are learned during the training process, granting CNNs greater flexibility in learning more complex and specific features of the data. This capacity to learn a vast variety of spatial features, coupled with the feature of weight sharing, is what makes CNNs a powerful tool. Figure 2.4.4 shows an example of how an activation is computed for a particular layer in a CNN. The kernel K in this diagram represents the shared weights which learn a visual feature. This feature then shows various levels of activation for a particular region of the input. We represent the convolution operation as a $*$, this performs multiplication per unit of overlapping space then adds the responses from these multiplications and adds them resulting in

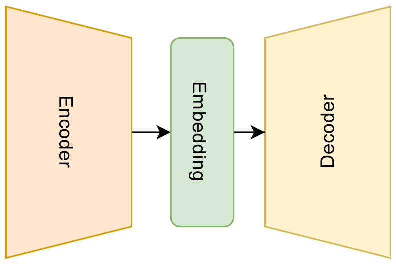


Figure 2.5.1: Basic structure of an Auto-Encoder Network. For the most part these are bottlenecked networks, meaning that the solution is undercomplete, however there exist sparse auto-encoders with overcomplete solutions.

a per feature activation map in the output. Given an input matrix I of size (m,n), a kernel K of size (f,f), a stride s, and padding p, the convolution operation can be expressed as:

$$\text{output}[i,j] = \sum_{u=0}^{f-1}\sum_{v=0}^{f-1} I[i\cdot s+u-p, j\cdot s+v-p]\cdot K[u,v] \tag{2.4.17}$$

Here $I[i\cdot s+u-p, j\cdot s+v-p]$ refers to a specific coordinate in the input matrix, taking into account the current position of the kernel (given by indices i,j), the stride s, and the padding p. $K[u,v]$ refers to the corresponding coordinate in the filter or kernel, at indices u,v and the summation over u and v computes the weighted sum of the overlapping region between the input and the kernel at the current position.
These convolutional operations can be stacked in many many layers to get feature maps of feature maps for more and more detailed image processing [50].

2.5 Deep Learning Architectures

Deep Learning (DL) is a paradigm in machine learning which is simply an extension of the concept of the MLP introduced earlier. If an MLP extends a perceptron by introducing several layers of them, deep learning generalizes this concept and extends it to other network architectures such as the CNN or recurrent networks. This section largely focuses on the application of deep learning to learning representations of input in a *self-supervised* manner. Understanding the concept of an auto-encoder is central to the work of this thesis, this section dives into the basic concept as well as a few variations.

2.5.1 Auto-Encoder Network Architecture

The idea of a bottlenecked network to learn lower dimensional representations of inputs is not a novel one [54], however when referring to auto-encoders in this thesis, unless otherwise specified, the implication should be that of convolutional auto-encoders. The auto-encoder consists of two distinct segments, an encoder block and a decoder block. The encoder takes an input and maps it to an intermediate representation, then the decoder takes the intermediate representation and attempts to reconstruct the original data from it. Figure 2.5.1 shows this structure in abstract terms. In its most basic form (one input layer, a bottleneck layer and an output layer) the auto-encoder architecture can be equated to PCA with somewhat weaker mathematical assurances regarding the found linear combinations of features. Auto-encoder architectures become powerful when we consider that they, in fact, represent a generalization of this structure, and are able to learn non-linear combinations of inputs as well, providing for much more flexibility.

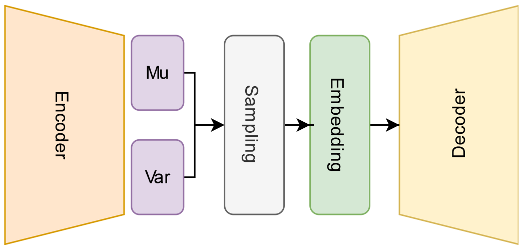


Figure 2.5.2: Basic structure of a variational auto-encoder Network. The activation of the last layer of the encoder is used to estimate a mean and variance for the sample passed in, this mean and variance are then sampled to generate an embedding from which the decoder then works.

Because this is a *self-supervised* learning method, it means that meaningful information can be extracted so long as data is available, even without the presence of explicit labelling. Given a bottlenecking constraint, we may learn useful features about the data in addition to creating a compression of the data. In the case of an overcomplete solution with sparsity constraints we would still learn useful features of the data in the representation. In this case, this representation that we pick out from one of the hidden layers in the network (typically the middle one with fewest units) is our latent representation. An interchangeable term for this may be an embedding. All told, a collection of these embeddings for an entire set of data may be referred to as subspace or latent space.
These representations created using an auto-encoder composed of fully connected layers is missing one key element that we are looking for, however. To make the leap from merely useful representations to semantically meaningful ones, one must consider the input as more than just a collection of simple features, one must consider them as features in a particular context. Convolutional layers as described earlier do this quite well for visual features. Various recurrent networks do this well for sequences as well. Once we have representations of the data, be they semantic or otherwise, the embeddings may be amenable to analysis using other unsupervised methods to further learn the structure of the underlying data and extract meaningful relations between the data points.

2.5.2 Variational Auto-Encoder

Variational Auto-encoders (VAEs) constitute a class of generative models that have gained considerable prominence in the domain of unsupervised learning. VAEs fuse the principles of deep learning with Bayesian inference to facilitate the learning of complex data distributions, making them particularly valuable for tasks such as anomaly detection, image generation, and even reinforcement learning.
VAEs extend the traditional auto-encoder architecture by incorporating a probabilistic spin. An auto-encoder generally consists of two primary components: an encoder, which transforms the input data into a latent representation, and a decoder, which reconstructs the original data from this latent space. In the case of VAEs, the encoder generates not just a fixed point in the latent space but a distribution, specifically, parameters of a Gaussian distribution. The decoder then generates the output by sampling from this Gaussian distribution, thereby introducing a stochastic element. See Figure 2.5.2 for an visual representation of how the variational auto-encoder works as compared to the earlier presented basic auto-encoder.
The objective function of a VAE is composed of two parts: a reconstruction loss, which encourages the decoded output to resemble the original input, and a regularization term, which forces the latent distribution to align closely with a standard Gaussian. The balance between these terms determines the quality of the generated samples and the structure of the latent space.

While the benefits of VAEs, such as the structured latent space and the ability to generate new data, make them appealing for various applications, it is essential to consider their limitations as well, including the potential for blurry generated images and the need for careful model tuning and training. This blurriness comes from the way the latent embeddings are sampled (where multiple inputs may map to identical samples). At the same time, this ability to explore the latent space by sampling the generated distribution makes it ideal as a generative model and gives the model some degree of explainability.

2.5.3 Sliced Wasserstein Auto-Encoder

Sliced Wasserstein Auto-Encoders (SWAEs) [55] represent a novel variant of auto-encoders, innovatively leveraging the principles of optimal transport theory, particularly the Sliced Wasserstein distance, to establish a more robust and meaningful latent space representation compared to VAEs.
Unlike VAEs, SWAEs work more like standard auto-encoders meaning that an input is mapped directly to a latent variable without an intermediate distribution modelling and sampling step. Then instead of matching the sampler distribution to a prior, we match the aggregated posterior of the latent space with a prior distribution. Typically this consists of a standard Gaussian or uniform distribution.
The Sliced Wasserstein distance provides a computationally efficient and differentiable means to compare distributions, overcoming the computational challenges often associated with the Wasserstein distance. It operates by projecting the distributions to multiple 1-dimensional subspaces and comparing these projections using the 1-dimensional Wasserstein distance, which can be computed efficiently.
The objective function for a SWAE thus incorporates a reconstruction term, similar to VAEs and other AEs, and a regularization term based on the Sliced Wasserstein distance. The equation for this is given in 2.4.11.
The adoption of the Sliced Wasserstein distance empowers SWAEs with several appealing qualities, including the ability to avoid the blurriness often associated with VAEs and the formation of a more interpretable and better-structured latent space. Nevertheless, the application of SWAEs necessitates careful consideration of the choice of prior distribution and the computation of Sliced Wasserstein distance, requiring a level of understanding of optimal transport theory.

2.6 Clustering

Many methods of self-supervised learning then proceed to use quite basic means such as clustering in order to evaluate their validity. The high-dimensional sub-space where all the embedding of these self-supervised methods happens is very conducive to return to these methods given that we've now gone from data where features may appear in various parts of the observation to a feature space where features are once again always in the same position in a vector. To this end, we cover two clustering methods later applied, k-means; a very scalable and fast method and Gaussian mixture modelling; something which is able to much better model multivariate data, however struggles computationally in high dimensions.

2.6.1 k-Means

k-means clustering is a method for unsupervised data partitioning commonly used in data analysis tasks [56]. It applies vector quantization in order to learn a set of points which are prototypical of groupings within our data, these are then also used to create a partition of the data. To this end, the algorithm applied for computing the cluster means is called Lloyd's algorithm and it repeats two steps until convergence [57]. After initializing the starting k points from which we will contine with quantization [58], we can start with the iterative refinement process. Given k means denoted as m_i where $i \in \{1,2,\dots,k\}$ we then assign points to these centres where point x_p is assigned to cluster C_i if $D(x_p, c_i)$ is smaller than any other $D(x_p, c_j)$ where $j \neq i$ and $j \in \{1,2,\dots,k\}$. $D(x,y)$ here simply denotes the distance metric between points x and y, in this case, this is the euclidean distance given by $D = ||x-y||^2$. Once the assignment stage is complete, set C_i contains all the points in our data which are closest to m_i according to our distance metric. Next we compute the new mean for each of these clusters by:

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in C_i^{(t)}} x_j \tag{2.6.1}$$

Intuitively all this means is that in every iteration we first assign points to their closest mean, then using these sets of points that we have just created, calcualte the new mean of the set and set that as the new mean. This continues until convergence which can be conceptualized in one of two ways. First we can imagine it as minimizing the total distance between every point and its closest prototype in the whole set. Alternatively one can imagine that when the prototypes move less than a certain amount between iterations, or not at all, the algorithm has converged.
k-means is unfortunately rather susceptible to the curse of dimensionality problem, however [59, 60]. The curse of dimensionality is a phenomenon specifically of the interaction of certain algorithms and their use of the squared euclidean distance as a comparison metric. The greater the number of dimensions that our dataset appears in, the less distinct comparisons become. An appropriate solution, for example would be the application of the generalized Minkowski metric with a normalization term in the interval $[0,1]$ given that the higher the higher the norm, the faster this problem comes to a head.

2.6.2 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are a type of probabilistic model commonly used for clustering and data representation tasks [61]. Unlike k-means, GMMs employ a soft assignment strategy for data partitioning, which allows them to handle complex data distributions more effectively. They accomplish this by assuming that the data are generated from a mixture of several Gaussian distributions, each characterized by its own parameters - mean (μ) and covariance matrix (Σ).
The GMM is represented as a weighted sum of M component Gaussian densities as shown in the following equation:

$$p(\mathbf{x}) = \sum_{i=1}^{M} \pi_i \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) \tag{2.6.2}$$

where π_i are the mixing coefficients satisfying $\sum_{i=1}^{M} \pi_i = 1$ and $\mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i)$ denotes the i-th Gaussian density function. Each of these component Gaussian densities contributes to the overall model proportionally to its mixing coefficient, allowing the GMM to approximate diverse and complex data distributions.
The parameters of a GMM, namely the set of means μ_i, covariance matrices Σ_i, and mixing coefficients π_i, are typically estimated from data using the Expectation-Maximization (EM) algorithm [62]. This iterative algorithm optimizes the log-likelihood of the observed data given the parameters, and consists of two steps: the E-step (Expectation) and the M-step (Maximization).
In the E-step, the posterior probabilities $p(z_i = 1|\mathbf{x})$ (also known as responsibilities) are computed, where z_i is a binary random variable indicating the component to which observation $\mathbf{x}$ belongs.

In the M-step, the model parameters are updated using the current responsibilities. The updates are performed such that the expected log-likelihood of the complete data (observed data plus latent variables) is maximized.
The EM algorithm is repeated until the parameters or the log-likelihood converge, typically yielding a locally optimal solution. An advantage of GMMs is their ability to model different shapes and sizes of clusters due to the flexibility of the Gaussian distribution. However, they can suffer from overfitting in high-dimensional spaces, which can be mitigated by using regularization or choosing appropriate covariance structures.

2.7 Clustering Validity Evaluation

Given that the general idea of this thesis is to create a reasonable embedding subspace for spectral data in an unsupervised manner, there needs to be a way to evaluate this portion of the work in a similarly unsupervised manner as well. Clustering on the latent space in order to test for density estimates in this space is not a novel idea [63]. For the most part these methods all stem from the same basic conceptual idea. Calculating the distance of points within a cluster or a partition and then comparing that to the distances between points outside the cluster of interest. Through this several metrics have been defined which can say something about the quality of a partitioning scheme [64]. For other clustering methods, such as Gaussian mixture modelling, there exist a few intrinsic methods of evaluation which can be useful to find the optimal number of components. Several of these methods are covered in this section as well.

2.7.1 Silhouette Score

According to [64] the Silhouette score [65] is the most robust metric available for evaluating the partitioning scheme of basic clustering models like k-means. It is hardly surprising then that it is also very widely applied then on simple problems where k-means is able to perform adequately. The Silhouette score for a point i in a dataset is given by s_i

$$s_i = \frac{b_i - a_i}{max(b_i, a_i)}$$

where;

$$b_i = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j_k} d(i, j)$$

$$a_i = \frac{1}{|C_i| - 1} \sum_{j_i, i \neq j} d(i, j)$$

Here b_i is the inter-cluster distance, meaning the average of the distance of point i to all points in the closest cluster which it does not belong to and a_i represents the intra-cluster score; so the average distance of the point to all the points which belong to the same cluster as i.

2.7.2 Bayesian Information Criterion

The Bayesian Information Criterion (BIC), also known as the Schwarz Information Criterion [66], is a widely used criterion for model selection among a finite set of models in the realm of statistical

inference and machine learning. It is based on the principles of Bayesian probability and provides a measure of the trade-off between model complexity and goodness of fit.
BIC is defined mathematically as:

$$BIC = 2ln(L) + kln(n) \tag{2.7.1}$$

where L represents the maximum value of the likelihood function for the model, k is the number of parameters in the model, and n denotes the number of observations or samples. The first term, $2ln(L)$, is a measure of the model's fit to the data, with a smaller value implying a better fit. The second term, $kln(n)$, penalizes model complexity, discouraging overfitting by adding a penalty that increases with the number of parameters.
In the context of model selection, the model with the lowest BIC is preferred. BIC's unique feature lies in its asymptotic consistency, meaning that as the sample size increases, the probability of selecting the true model also approaches one, assuming that the true model is within the set of candidate models. Despite its ubiquity, BIC does make an assumption of a large sample size, and as such, may not be as effective with smaller datasets. Additionally, the strict penalty term based on the number of parameters may not always be suitable for all situations, particularly when a more complex model is justifiably needed. Nonetheless, BIC remains an invaluable tool for model selection in a variety of statistical and machine learning contexts.

2.8 Simple Classification

2.8.1 Gaussian Naive Bayes Classifier

The Gaussian Naive Bayes (GNB) classifier [67] embodies an integral application of probabilistic classification methodologies in the realm of machine learning and data mining. This classifier capitalizes on the fundamentals of Bayes' theorem in conjunction with an arguably simplistic assumption of feature independence. The selection of GNB as a classifier is often predicated on its computational efficiency and ease of implementation, thereby making it an apt choice for high-dimensional data analysis. It works by ascertaining the conditional probability of each class given a sample, with the class yielding the highest probability being selected as the prediction.
Inherent in the GNB model is the presumption that the continuous values linked with each class conform to a Gaussian or normal distribution. Characterized by its symmetry and bell-like shape, a Gaussian distribution is delineated by its mean μ and standard deviation σ, which respectively dictate the central tendency and dispersion of the data. This assumption, while enabling a facile implementation of classification tasks, may not invariably be valid. Cases where the distribution deviates from Gaussian, or where the features are not mutually independent, may lead to a suboptimal performance of the model. Nonetheless, the inherent simplicity, scalability, and adaptability to high-dimensional datasets underscore the widespread application of GNB classifiers, despite potential constraints.

2.8.2 Random Forest

Random Forest [68] is a potent and highly adaptable machine learning methodology that further enhances the utility of decision trees. As an ensemble algorithm, it amalgamates the predictive capacity of several models, specifically decision trees, thereby optimizing the overall model performance and stability. It demonstrates exceptional prowess in countering the prevalent issue of overfitting inherent in single decision trees.
The element of ”randomness” in Random Forest emanates from two distinct aspects. Firstly, every decision tree in the ensemble is constructed on a bootstrapped sample of the original dataset, a sample

drawn with replacement, adhering to the concept of bagging (bootstrap aggregating) with an intent to mitigate variance. Secondly, at each decision tree node, a randomized subset of features is chosen to discern the optimal split. This process effectively de-correlates the trees, augmenting the robustness of the forest against the potential bias of a single dominant feature. The terminal prediction in a Random Forest classifier is typically realized through a majority voting mechanism across the individual decision trees for classification tasks.
Despite its relative simplicity and limited tuning parameters, Random Forest classifiers are revered for their high predictive accuracy, resilience, and user-friendly nature. They are capable of processing both numerical and categorical data, adept at handling missing values, and furnish an estimate of feature importance. These qualities render them a go-to machine learning algorithm across a broad spectrum of applications in both academic research and commercial industry sectors.

2.9 Related Work

There is a previous iteration of this research focusing on variational auto-encoders in order to create an inspection method for radio astronomical data [18]. The research proposes a variational auto-encoder which takes two separate inputs of the phase and amplitude information, then uses a concatenation of both as a latent representation. In order to avoid the lengthy process of labelling LOFAR data, in this research the authors instead elect to construct a proof of concept system on HERA [19] data, both real and simulated. The research notably uses interpolation in order to fit the data to an auto-encoder's input constraints. In our research we choose to forego this in favour of patching which is a different way to deal with this problem. Both methods ostensibly face challenges when it comes to data of multiple time-scales, however we believe that [18] were somewhat more selective of their data beforehand.

Chapter 3

Methods

Building on the concepts explored earlier in the theoretical background, this chapter collects those theoretical components and presents a novel method for learning embeddings of spectral data as well as an evaluation framework to test it. First the data format is explained alongside creating a dataset to evaluate tasks against labelled data. Second, several representation learning methods are discussed, including the justification and construction of the central model in this thesis. This model as well as several other 'baseline' models are implemented and their training processes detailed. Lastly a series of tasks and evaluation metrics are defined which will give insight into the models' empirical performance for this particular domain. For a clearer overview of this process, see the pipeline diagram in Figure 3.0.1.

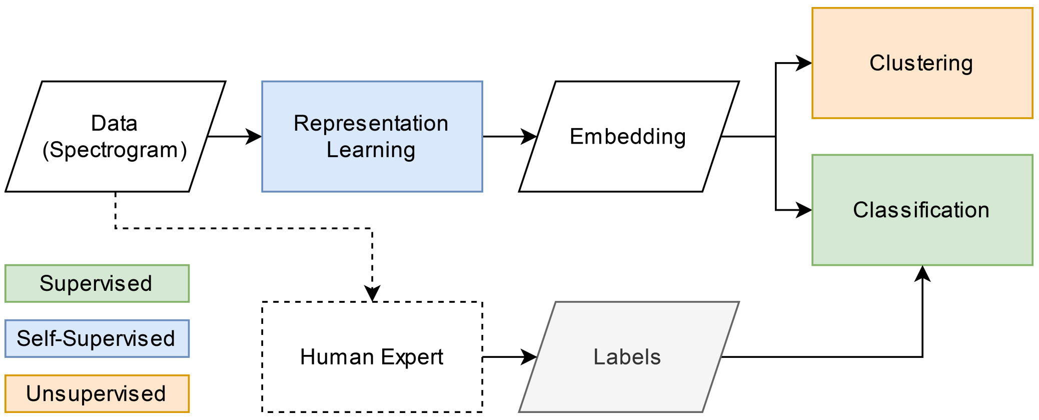


Figure 3.0.1: Proposed full pipeline from data to tasks that the components are evaluated on. Note the partial reliance on a human expert. This is the portion of the pipeline which this project aims to bypass as much as possible in order to streamline data processing, however some human labelling is required to kick-start the process.

3.1 The Dataset

The LOFAR telescope collects and records data in very large streams when making observations. Each station records the signals from tens of antennas at up to 200 MHz at the station level. This results in very large observations consisting of several TB of time/frequency data in both the real and imaginary domains. As a byproduct of this data collection, a downsampled version of these observations is produced for long term availability and archiving. This is the data that this project

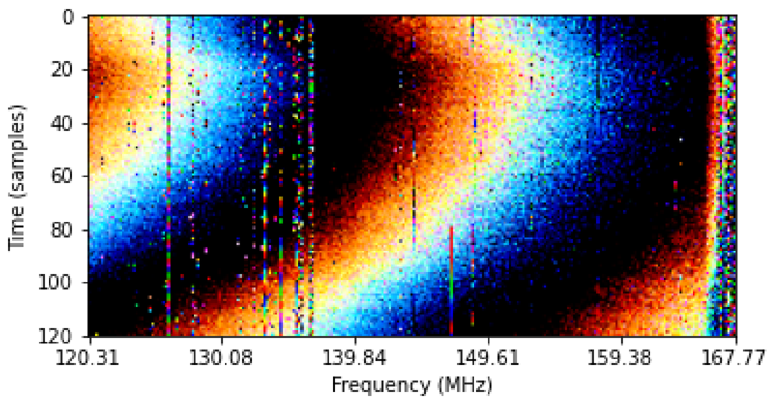


Figure 3.2.1: An example of a baseline spectrogram in phase format. Note that the frequency is on the x-axis and time is on the y-axis. This is done purely out of convenience for use with the labelling software. Using 'tall' images made them harder to discern in the software. Any other images in this format should be assumed to be in the same orientation.

aims to work with as it is on a more manageable scale and appropriate for most detection tasks and proof of concept implementations. As part of normal operation, some data tagging is performed at the observation level. For the purposes of this project these pre-existing labels are somewhat too broad, however. In the hierarchical structure of the observations, the labels are applied at observation level, where this project attempts to look at the data at a more granular per-spectrogram level. To facilitate this, in addition to the learning methods and inspection tools, this project delivers a dataset of about 4000 multi-attribute labelled samples. These examples come from the aforementioned real down-sampled data. This labelled dataset is not used for training, but evaluation only. This results in the ability to evaluate an un-supervised learning method with labelled data.

3.2 Representing Complex Data as an Image

The LOFAR telescope, as far as the data selection in this thesis is concerned, produces complex valued data in 4 channels. Here a method is proposed for mapping these to an image in the RGB space which is useful for a labelling task or similar. In this case the complex signal can be decomposed into two distinct parts, the phase and the magnitude of the signal at each point. While both are informative about the data in certain ways, the phase infromation is picked here as the definitive, because it produces more distinct patterns which can be inspected. For the 4 polarization observations extracted from the set of LOFAR data, the mapping of these 4 channels to RGB is a challenge. This is done with the following formulas; $R = c_{xx}$, $G = 0.7c_{xy} + 0.7c_{yx}$ and $B = c_{yy}$ where for example c_{xy} represents the channel of the observation for the 'xy' polarization. For an example of how the data looks when mapped using this method and a few examples of the label classes which will be described later, see Figure 3.3.2. See additionally Figure 3.2.1 for details on how the phase spectrograms are laid out with regards to frequency and time axes.

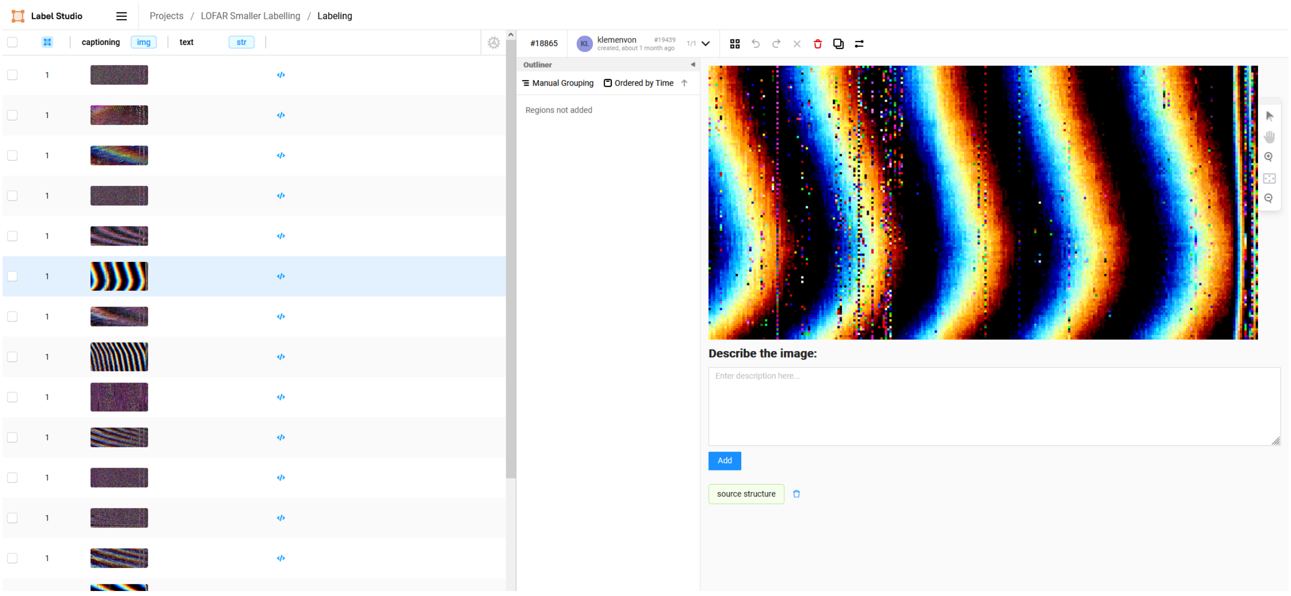
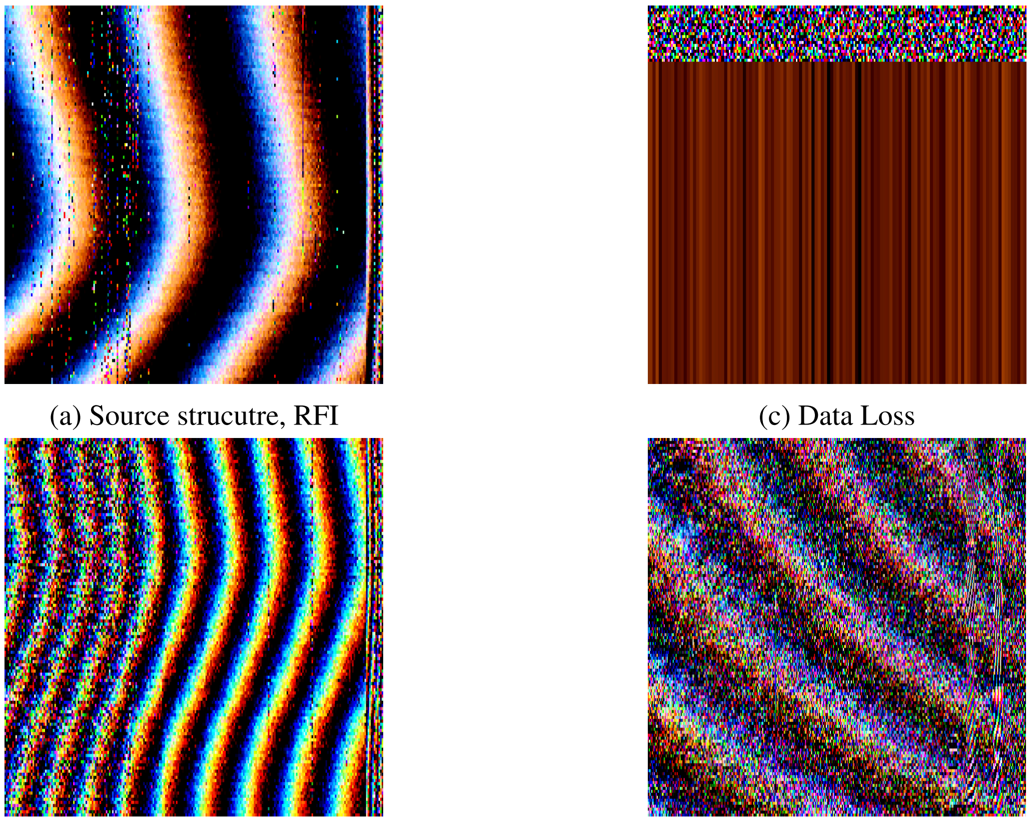


Figure 3.3.1: Image of the label studio instance used during labelling. The example image shows a partially labelled sample. Of the labels defined later in this chapter, we would assign here 'source structure' and 'narrow band RFI'. (see the vertical lines in the extreme right and middle left of the spectrogram)

3.3 Data Labelling

The full set of LOFAR observations includes 4̃,500,000 baselines in 1889 sub-array pointings across 371 observation files. However, a large majority of these are baselines recorded in very few frequency bands. Too few to constitute a good spectrogram image for our considerations. For the most part these are calibration observations. While these are likely to provide good information, we would like to consider larger input patches in this study where a proof of concept is explored. Given certain base requirements for data size (at least 90x90 in the time and frequency domains respectively and at least 4 polarizations) about 10% of the dataset is retained with 4̃40,000 valid baselines in consideration. Of these 4438 baselines are taken as a subset (2 observations) and labelled. The annotations in this case come in the form of a list of attributes assigned to a particular baseline based on visual features. In order to expedite this process, the open source labelling tool Label Studio is used for attribute labelling. Making a segmentation dataset is not in the scope of this study, so the smallest unit of data considered here is a single baseline spectrogram. See Figure 3.3.1 for reference of the labelling interface and process.
Leveraging the expertise of some frequent LOFAR users, several key label attributes are determined. Some are simply descriptions of the visual patterns which are normal such as *source structure* and others are attributes describing anomalies, such as *data loss* or *decorrelation*. A full account of the label classes developed is given in Table 3.1 below. These labels span baselines in 2 sub array pointings across 2 observations.



(a) Source strucutre, RFI

(c) Data Loss

(b) Source structure, Decorrelation , RFI

(d) Broadband RFI

Figure 3.3.2: Showing four sample observations as mapped to RGB by the scheme defined in Section 3.2. Examples show some characteristic samples of the labels that will be expanded on in Section 3.3. Figure 3.3.2a shows an example of a relatively clean observation with strong source structure (low frequency background feature) additionally showing some Radio Frequency interference patterns between 127 and 138 MHz. Figure 3.3.2b shows characteristic data for an example of strong scintillation shown by the noisy region around 120 MHz. Figure 3.3.2c shows an example of data loss, this is additionally an unusual sample as it presents data loss intermittently rather than dropping out entirely until the end of the observation. Finally Figure 3.3.2d shows an example of broadband radio interference along with source structure and generally a noisy background. This differentiates, rather obviously, from the narrow band RFI shown earlier as it occupies and interrupts a wider range of frequency bands.

Attribute	Type	Near/Far	# Examples
Source Structure	Feature	Far	3109
Broadband RFI	Anomaly	Near	1159
Checkerboard Artifacting	Feature	Far	565
Data Loss	Anomaly	Near	339
Decorrelation	Anomaly	Near	173
Autocorrelation	Feature	System	128
Vertical Artifacting	Anomaly	Unknown	35

Table 3.1: All of the labelled examples available. Shown is the attribute or label, the type of feature it represents (not all features are anomalous) as well as whether this is a near or far field effect. Near field simply means that it is able to affect individual stations. And far field means that it is not a terrestrial interference and can therefore affect larger portions of the whole observation. Also listed are the counts per label class of available labels.

3.4 Data Preprocessing and Preparation

The requirements set out for the subset of ’usable’ data for this project do not state an exact set of dimensions which each baseline must have, only a minimum. This means that the data is not of uniform size, something which does not play very nicely with convolutional networks or independent component analysis. For this reason the data passed to the models comes in the form of patches taken from the data. These patches are of size 128x128, which is larger than the minimum required size of the observation. This means that any empty space making up the difference between the actual size and the patch size is zero padded. While this is possible to happen, both the training subset and the validation subset entirely avoid this and use observations which are larger than one patch. These patches have an overlap of 40% between them, meaning that some of the data is duplicated. This is done in order to ensure that as few critical features as possible are cut by the boundary of the patches. There of course remains the issue extremely high variance in the temporal scale. Considerations here include re-scaling or re-sampling the data, this is not preferable however. There is not enough data in the high data frequency observations to make a representative downsampling. On the other hand for the low data frequency observations, the data which has been dropped in the downsampling process is lost and irrecoverable, so no upsampling is possible here. The problem of different data scales is addressed at length in the results and later discussion.
In the selected training data and across most of the data which is available for this study, the frequency bins do not deviate in scale, staying rather stable at the LOFAR band-width of 195 kHz per bin. This is in contrast to the time scale where variations are quite common, but grouped into certain set lengths of observations. Observations do not necessarily share the same start and end points in absolute terms for the frequency bins either. Often they do, however this is not uniform over the data. This is first of all because of the different requirements of the observations and second of all because of the clear split between data recorded with the HBA and LBA sets of antennas. This means that by taking patches of this data, we are rather taking a ’pattern recognition’ approach in place of one based on an absolute coordinate system.
An aside that must be made here; because of the constraints of the ICA algorithm, training of which is described later in this section, the patches taken for training it are only 64x64 rather than 128x128. This is because the nature of the operations performed by ICA, namely singular value decomposition

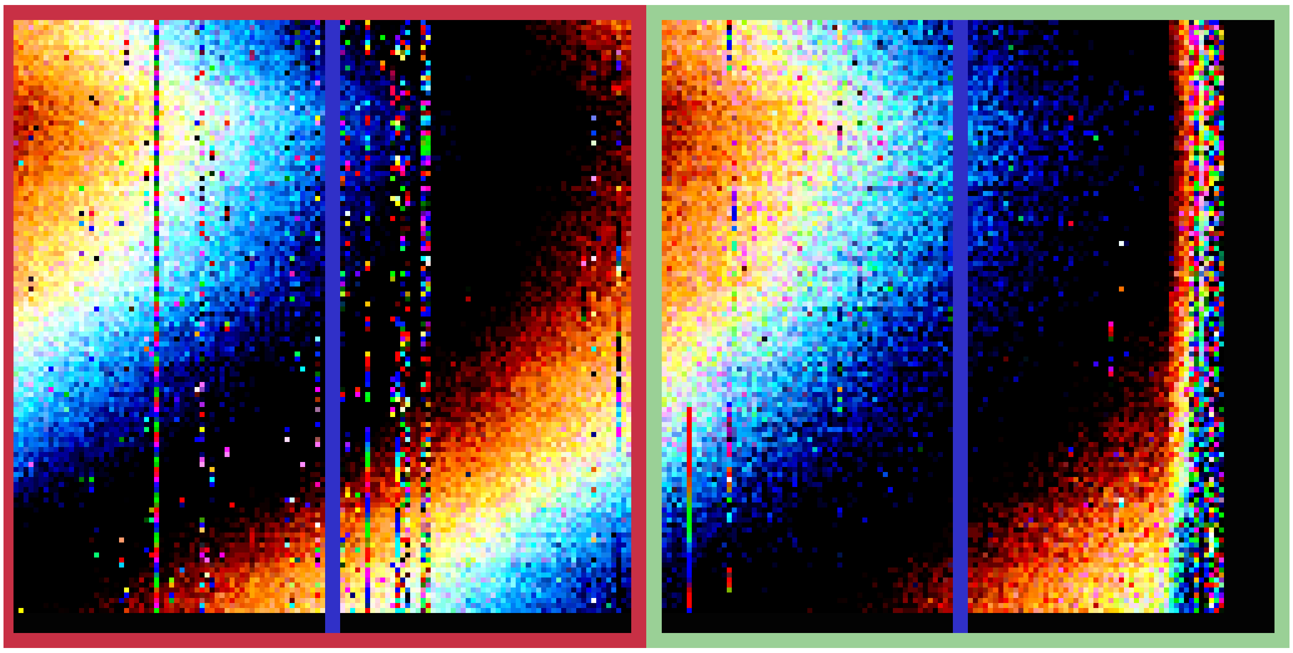


Figure 3.4.1: Example of how an observation is split into patches. This observation is originally too small in the time dimension to fit exactly in the 128x128 format, so the lower segment of the observation is padded with zero data. The process in this case results in 3 patches being taken with enough overlap to covered the data more than once.

scale very poorly with increasing matrix dimensions where computational costs are concerned.

3.5 Representation Learning

In this section, we go over the main component of this thesis, the representation learning algorithms used to embed the data into a sensible semantic embedding space. First presented is the baseline ICA method followed by several auto-encoder models including our proposed cascading auto-encoder.

3.5.1 ICA

Among the various methods applied to the problem, certain characteristics make the ICA approach distinct. One of the defining constraints of ICA is its hard orthonormality between the filters, which is combined with a highly localized way of finding these filters. This combination may result in a good likelihood of duplicate filters being discovered, a scenario that is not desired. Furthermore, the ICA's dependence on input size becomes a limiting factor, setting it apart as the least flexible method among those considered.

In the data preparation section (3.4), it was mentioned that other methods perform operations on a 128x128 patch taken from the data. Such a process is unsustainable with ICA, as it would necessitate computing an eigendecomposition for a 16384x16384 matrix at a minimum. Despite theoretical efficiency gains with linear algebra approximations, the solution in practice represents a time complexity $O(n^3)$, reducible only to $O(n^{2.376})$ with advanced techniques[69] (not applied here), making this approach infeasible at scale.

To mitigate this challenge, the input size is restricted to 64x64 patches. Although this compromise leads to learning smaller feature filters, it brings about more manageable computational requirements,

Hyperparameter	ICA	Only2D	SWAE	VAE	ProposedArch2D	ProposdArch1D
Input Size	64x64	128x128	128x128	128x128	128x128	128x128
Training Epochs	10000	200	200	200	200	200
Batch Size	150000	128	128	128	128	128
Latent Size	256	256	256	256	256	32
Optimizer	/	ADAM	ADAM	ADAM	ADAM	ADAM
Learning Rate	/	0.001	0.001	0.0001	0.0001	0.0001
Layer Activation	/	ELU	ELU	ELU	ELU	ELU
Last Layer Activation	/	Tanh	Tanh	Tanh	Tanh	Tanh
Kernel Size	/	3	3	3	3	4

Table 3.2: Parameter Table

with eigendecomposition for a 4096x4096 matrix in each iteration.
While ICA's hard orthonormality is effective at filtering degenerate solutions, it can lead to peculiar situations where the orthonormal solutions are not necessarily relevant. In fact, stable solutions often emerge where many of the ICA filters are learned combinations of noise. Therefore, reducing the number of output filters is advisable, aligning the solution with the undercomplete nature of the other presented algorithms.
A detailed list of parameters utilized for training this method can be found in the general parameters table 3.2. The fitting of the ICA model comes with its own challenges, requiring all data to be loaded into memory during each iteration. This constraint limits the training data to at most 150,000 64x64 patches in the matrix at any given time, assuming a 24GB system. The dataset sampling is guided by `torch.DataLoader`'s pseudo-random randomization function, and the data is segmented and accumulated by splitting the spectrogram channels, allowing for loading about 3000 baselines into memory for training. This amounts to just over 10% of the training set used with other methods. Although not ideal, the random sampling ensures representative data is chosen, and the system is able to fit just over one full observation into memory at once, eliminating the need for batch processing of observations. These practical considerations and their limiting factors are further discussed later.

3.5.2 2D CNN Auto-Encoder

The basic 2D CNN Auto-encoder is used as a template for constructing all the auto-encoder methods applied here apart from the second half of the cascading AE design. This is to keep the comparison later on as tightly focused on the latent space itself and how it was arrived at rather than the difference in encoding sizes.
Per-layer statistics for both the encoder and decoder parts of the model may be seen in Figure 3.5.1. Starting with an input size of 128x128 and a depth of 4 channels (one for each of the polarizations), the input is run in an alternating manner through a 3x3 kernel convolutional layer with stride 1 and then 2 in order to slowly reduce the number of nodes in each layer. For each encoder step, ELU activation is used in combination with batch normalization. Once we reach the final layer of the encoder, the result is flattened into a layer with 12,288 units which is then fed through a fully connected layer which reduces this further to 256 units. This is what forms the embedding. From the embedding, the process is reversed. From 256 up to 12,288 units, then re-shaped into a 3d stack of 2d filters. The same set of strides and activations are applied in the decoder with one notable exception. In the final

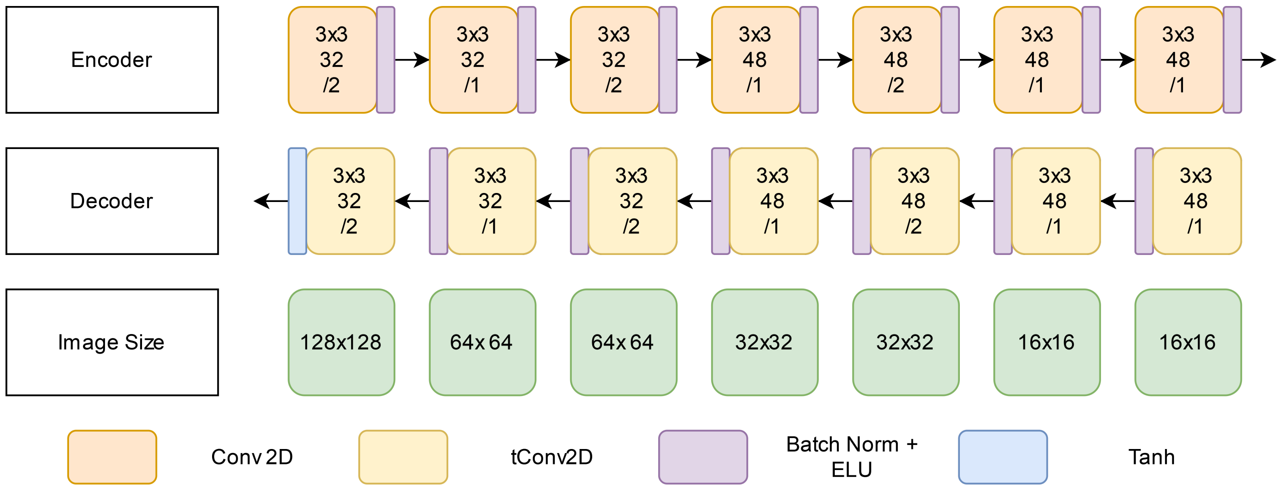


Figure 3.5.1: The basic encoder and decoder structures of every 2D auto-encoder used in the comparisons. The upper portion shows the encoder layers and the lower portion the decoder layers. For each of these, consider how they fit in the more general encoder-decoder structure shown in Sections 2.5.1 and 2.5.2 for auto-encoders and variational auto-encoders respectively. Shown in the diagram are convolutional layers in blue and batch normalization layers in purple. Also specified per layer are the kernel size, the number of filters and stride where stride is greater than 1. For example '3x3x48 /2' means a convolutional or deconvolutional layer with kernel size of 3x3, 48 filters and a stride of 2.

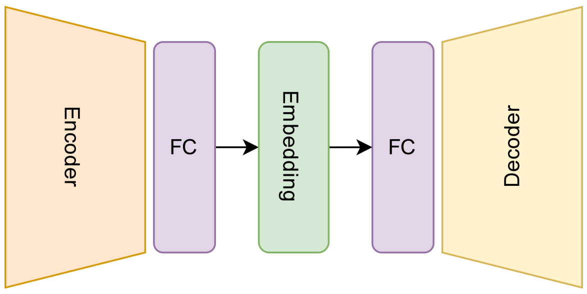


Figure 3.5.2: The structure of the basic 2D Auto-Encoder model with encoder and decoder structures described in greater detail in Figure 3.5.1

layer of the decoder the activation is changed to the *tanh* non-linearity in order to produce output bounded in the range of -1 and 1 such that the result ends up being a normalized image. The more generalized structure of the auto-encoder can be seen in Figure 3.5.2 where we see the overview of the structure comprising of the encoder, two fully connected layers bracketing the latent encoding and then the decoder.

3.5.3 Sliced Wasserstein Auto-Encoder

In structure the SWAE network is an exact match for the base 2D auto-encoder. The only difference is the additional loss term imposed on the latent embeddings. The distribution of the embeddings is enforced via the Wasserstein metric using a slicing trick in order to sample the distribution. The Wasserstein metric or more colloquially 'earth mover's distance' is described in greater in Section 2.4.4. Accordingly, as the 256 dimensions of the latent embedding remain the same, an appropriate number of projections to sample the latent distribution is selected at 50. This should give a reasonable estimate of the true latent distribution so that it may inform the enforced distribution given. In this

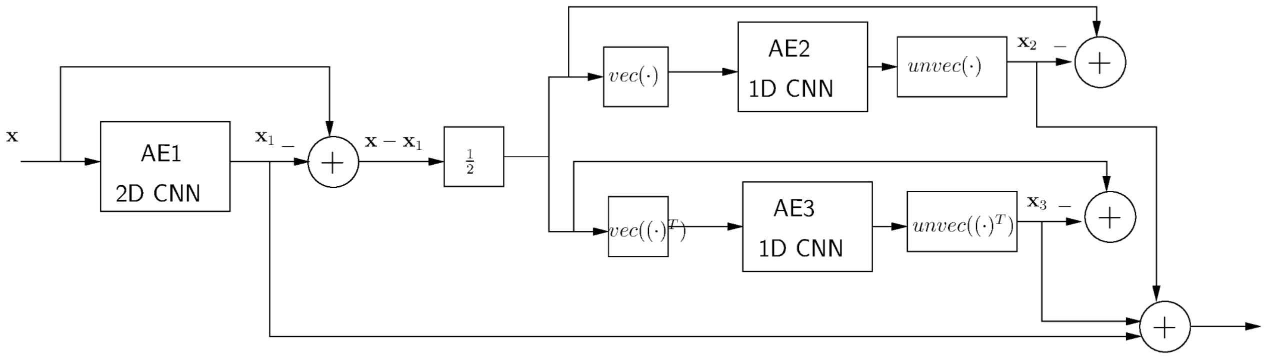


Figure 3.5.3: A diagram showing the structure of the proposed cascading auto-encoder method. The chart shows how the difference between the reconstruction and the original (aka. the residual) is taken, split in half by absolute value, then flattened in either direction for each 1D auto-encoder following the initial 2D auto-encoder. Diagram courtesy of S. Yatawatta.

case an N-variate Gaussian with a mean and variance of 1 centered around the origin of the embedding domain. Having selected our K sampling vectors we can formulate the additional sliced Wasserstein loss term using this. For each of these K vectors, the data is then mapped on them and represented as a quantile function of the univariate distribution of the data along k where $k \in K$. A single quantile function along k for either batch b or reference distribution r can then be represented as F_b^k and F_r^k respectively. Given this, the Wasserstein distance along k is then given by $SWD^k(b,k) = F_b^k - F_r^k$. Now for every k in set K of all mapping vectors, the total estimated Wasserstein distance for the set can be estimated. This method specifically affects the overall methodology and parameter choice in that the batch size is specifically enlarged for all involved deep learning methods in order to ensure better distribution estimates are enabled with the sliced Wasserstein metric giving a training batch size of 128 for all auto-encoder methods.

3.5.4 Variational Auto-Encoder

The second variant form of the basic CNN auto-encoder architecture is the variational auto-encoder which includes one additional fully connected layer in the space between the last feature map of the encoder and the first feature map in the decoder. This additional layer is there because the layer that creates the latent embedding for a particular item is composed of two heads. One estimating the means of the embeddings and the other estimating the log variances. This is described in more detail in the section covering vartiational auto-encoders in Section 2.5.2. A latent embedding is then sampled from this distribution and used for the reconstruction. This means that there is somewhat of a decoupling between the embedding created by the encoder and the one used by the decoder to make a reconstruction. While this allows for the generation of novel data that looks like the data initially trained on, the more desirable property here is the statistical map of the latent space. Namely the latent space is forced to fit an N-dimensional normal distribution. Where N is the number of elements in the latent space. The distribution is enforced by a second loss term on top of the means squared error loss. This additional loss term is the Kullback–Leibler divergence between the distribution of the data observed in a particular batch and a normal distribution in an equal number of dimensions. The KL-Divergene statistic is described in more detail in Section 2.4.4.

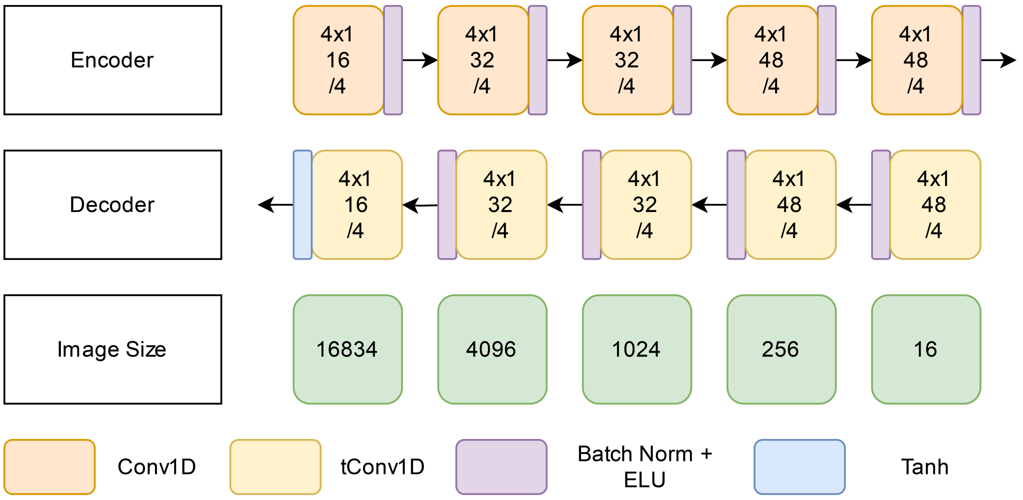


Figure 3.5.4: Detailed overview of the layers in the 1D auto-encoder components of the proposed architecture.

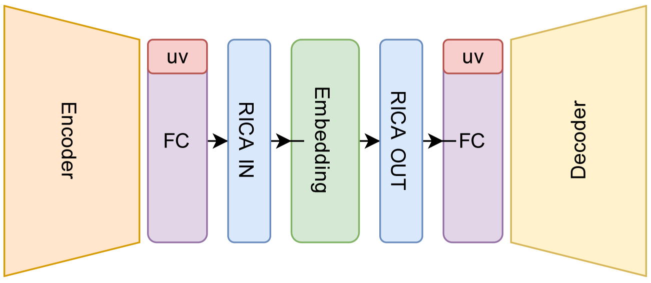


Figure 3.5.5: Overview of the encoder-decoder layers with the RICA regularization layers included. This also includes the image plane coordinates concatenated into the fully connected layers before the RICA input and output layers.

3.5.5 Proposed Architecture

Our proposed method consists of several parts. The 2D CNN auto-encoder structure is retained with minor modifications. The image plane coordinates of the baseline are embedded into the fully connected layers. This is achieved by concatenating the coordinates with the features before they enter the fully connected layer. The coordinates are then added again just before the decoder to become part of the reconstruction process.
Additionally, RICA is applied as normalization to the embedding layer to enforce a sparsity constraint, as seen in Figure 3.5.5. Despite this constraint, the embedding remains the same size as previous methods: a vector of size 256. The novelty in the signal processing domain is the introduction of two additional 1D auto-encoders in a cascade.
As shown in Figure 3.5.3, input to the 1D auto-encoders is generated by taking the difference between the original image and the reconstruction. This difference is then halved and flattened separately in the time and frequency domains. It produces 1D vectors of size 16384, resulting from unfolding the 128x128 patch. Two directionally unrolled patches are passed through the cascaded 1D auto-encoders. These 1D auto-encoders are convolutional with a kernel of size 4. Their structure is similar to the 2D auto-encoders but varies in the number of layers in the encoder and decoder components. For the specific changes refer to Figure 3.5.4.
Each of the two 1D auto-encoders has a latent dimension of 32, also with a RICA-based sparsity constraint. Concatenating the latent space of all three auto-encoders results in a latent representation of size 320. Although this size is larger than comparable methods, the RICA-imposed sparsity constraint necessitates this larger latent space to encode the data effectively.
There are two training choices: end-to-end or cascading. We chose cascading as it leads to more stable training. End-to-end training could result in the 1D auto-encoders not learning well, as changes in the 2D auto-encoder might present them with a moving optimization target.
The same adaptations are made in the 1D auto-encoders as were proposed earlier for the 2D auto-encoder. Image plane coordinates from LOFAR are taken and concatenated with the flattened output of the last 1D convolutional layer before it is passed through RICA. The two additional 1D auto-encoders aim to pick up strong features in the frequency domain (RFI) or the time domain (data loss) to enhance our reconstruction and embedding.
Comparable to the VAE's KL-Divergence term, RICA is very sensitive affecting the chosen learning rate for this approach as well. As such, VAE and our proposed architecture both operate with a learning rate of 0.0001. Overall the training is also twice as long. Because we train in a cascading manner, the 2D auto-encoder is trained first for 200 epochs, then the weights are frozen and the two 1D auto-encoders are also trained for another 200 epochs resulting in a total training regimen double that of comparable methods.

3.6 Tasks

3.6.1 Unsupervised Clustering Evaluation

First, we borrow techniques from data-mining as a precursor to other evaluation methods. This is due to labels only being applied to a few observations, with none reserved for the training set. The evaluation's first step will be to cluster the found embeddings and try to determine the number of clusters in the data. We will do this using two separate clustering methods: k-means and GMMs.
K-means is highly scalable for large datasets and can test for nearly any value of k. In contrast, GMMs are less scalable. Using them on embeddings of the intended size is near the limit of what can be fed

to GMMs. Expecting a converged model within a reasonable time frame may be challenging.
Following the actual clustering each of the partition schemes obtained from the clustering process is evaluated. For k-means the most appropriate method is the Silhouette score [64] and for Gaussian mixtures, the Bayesian information criterion will provide the basis for evaluation. While both the Akaike information criterion and the Bayesian information criterion are appropriate for this task [70], the Bayesian information criterion is picked as it much more strongly penalizes the model complexity. This is a desirable property as we are trying to find the least number of mixtures to decompose the dataset to.

3.6.2 Classification

Simple classification methods such as Random Forest and Naive Bayes classifiers, referred to in Sections 2.8.2 and 2.8.1 depend quite heavily on the quality of the features of a particular dataset for quality classifcation. We are able to effectively leverage these methods in order to evaluate our model's embedding potential by evaluating it through the embedding's ability to support simple classification. The latent spaces generated by our models have comparable dimensions, but ICA's pre-processing is dissimilar to the other methods. The smaller patch size means that it will generate more embeddings per correlated baseline. For this reason, we compare the methods using two separate classification methods. This additionally allows us to evaluate methods against a classifier's base assumptions for further comparison.

3.6.3 Visual Inspection

Lastly, as one of the stated purposes of this research is to create a method which presents a user with a 'zoo of artifacts' where similar items are grouped together thus creating clusters or regions of space where certain features in the data all appear together, visual inspection is also an option. For this, we leverage the power of t-SNE as described in Section 2.3.3 which is able to take data in an arbitrary number of dimensions and map it to 2D in such a way that items in close proximity in N dimensions also appear close in 2D. Neither this method nor simple visual inspection are robust enough to serve as an empirical means of evaluation, however this can simply serve as the additional confirmation of the results which come out of the classification stage. For this purpose we develop a tool which is able to display side by side a t-SNE plot of the embedded data, then display elsewhere the images associated with each of the points in the embedding as points are selected. This ability to explore the data in an interactive manner should serve as invaluable in research where the empirical evaluation is such a challenge.

Chapter 4

Results

In this section, an overview is given of the model training for each of the models presented previously. Later in the section, we move on to model evaluation through the tasks which are defined in Section 3.6. The groundwork is laid for comparing the ICA model to the auto-encoder methods, then the auto-encoder methods are compared by their common elements and evaluated on the disparate features of their methodologies. We provide a brief overview of the clustering as well as the validity indices which attempt to find the optimal latent space layout for each method. Later a slightly more in-depth evaluation making use of Random Forest and Gaussian Naive Bayes classifiers is presented as a potential downstream task and a method of evaluating the quality of the embeddings. Lastly a manual inspection is made of the latent space as described by a t-SNE embedding, this is done with an application developed for this work which is able to show images of corresponding points in latent space in order to evaluate the latent embedding's suitability for use as a data-inspection intermediary. The findings presented in this section are then further discussed in the final chapter.

4.1 Model Training

Models are trained on a subset of the LOFAR downsampled visibilities dataset comprising of 22500 baselines except for ICA (see Section 3.5.1) which is trained on a further pared down randomly selected subset of the training set. The deep learning models are trained until the maximal epoch without early stopping. ICA is trained with the possibility of early stopping, however the data complexity and the very low convergence threshold means that the method never converges. Lack of convergence in this case puts it in line with the auto-encoder models which are also trained without early stopping until 200 (or 400) epochs. Concerning the comparison of the auto-encoder models the plots in Figure 4.1.3 reflect only the Mean Square Error (MSE) or reconstruction loss as this is the only common element between all the networks trained. This shows the reconstruction performance of each of the models throughout the epochs of training. Other loss categories which differ are not comparable and therefore not shown explicitly here. The potential problems of dual-target optimization are weighed against the favourable constraints these additional targets force their respective models to adapt to. Primarily, the sections following the details of model training will attempt to discern if reconstruction quality or the additional constraints have the most favourable effects on the quality of the learned representations.

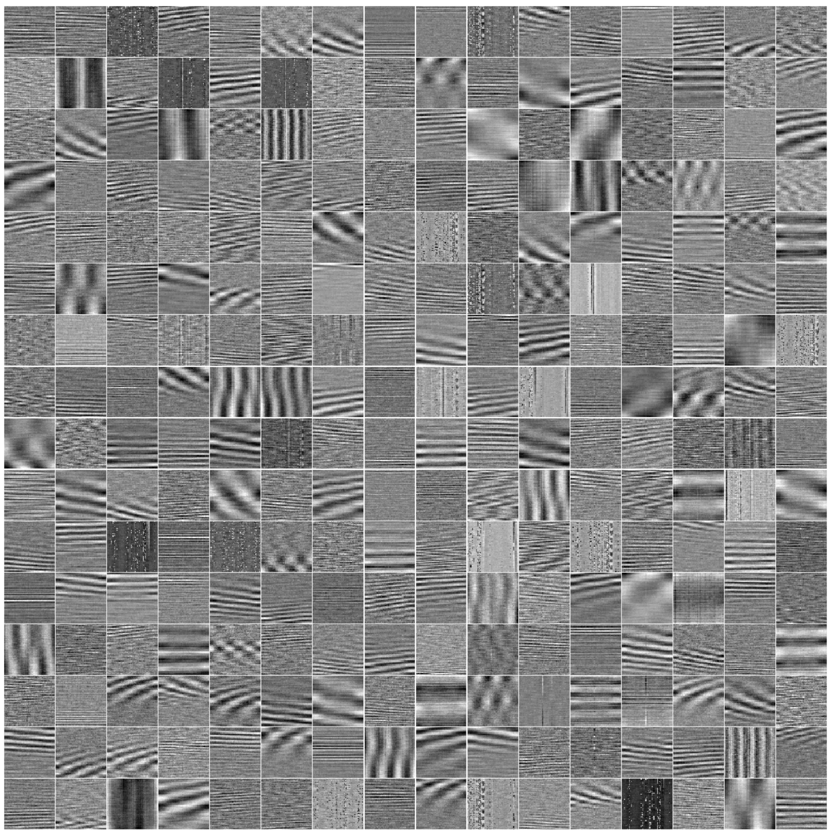


Figure 4.1.1: Learned ICA filters. Not ordered by eigenvalue importance. Seen are most of the major recurring patterns including auto-correlation related features as well as many features indicating source structure.

4.1.1 ICA

In this comparison study, we explore various methods, among which Independent Component Analysis (ICA) stands out as unique. Unlike other techniques examined, ICA does not rely on deep learning. Instead, it is a linear transform that has shown promising results on the given task, though not without some drawbacks, which we will discuss later.

ICA's working principle is straightforward: it creates a single set of filters that describe an input image through a linear combination. These filters can be directly accessed as a collection. To fit the ICA model, we used the same data-loader and pseudo-random order as for the deep learning methods, loading the first 3000 baselines into memory simultaneously. This process is roughly equivalent to handling 30 batches of data from a deep learning training perspective.

The result of this fitting process is a collection of 256 patch filters, each of 64x64 dimensions, as shown in Figure 4.1.1. An examination of this image reveals that these filters primarily represent features at various scales, rather than noise. An undercomplete ICA solution appears to work well in this context, as it avoids the high proportion of noise filters often found in complete solutions, a common problem when seeking a full solution or working with low-variance data.

One of ICA's major advantages over deep learning methods is its interpretability. The filters that comprise its component matrix are clearly displayed in the figure, allowing for easy visual inspection of the important features of the input. This contrasts with deep learning models, where understanding the significant features can be more complex.

4.1.2 Baseline Auto-Encoder Methods

Similar to ICA, the latent representation for all baseline auto-encoder based methods is restricted to 256 in order to create a consistent latent embedding size between methodologies. Similar reasoning is applied at least in this subset of methods to the input dimensions. ICA was the odd one out in this case as the input dimension of 128x128 was simply not possible while retaining a relatively sizeable training set. The latent size of 256 is already quite large, however during the course of a coarse parameter search, sizes up to 4096 were attempted, however for the most part; larger latent spaces tended to overfit much faster on the limited training set, therefore restricting this to 256 is appropriate in this case. In addition to this, all the methods in this section are trained for 200 epochs on the data. After this point, only the proposed architecture still has any further drop in validation accuracy, the rest of the models plateau earlier and begin to overfit after this point.

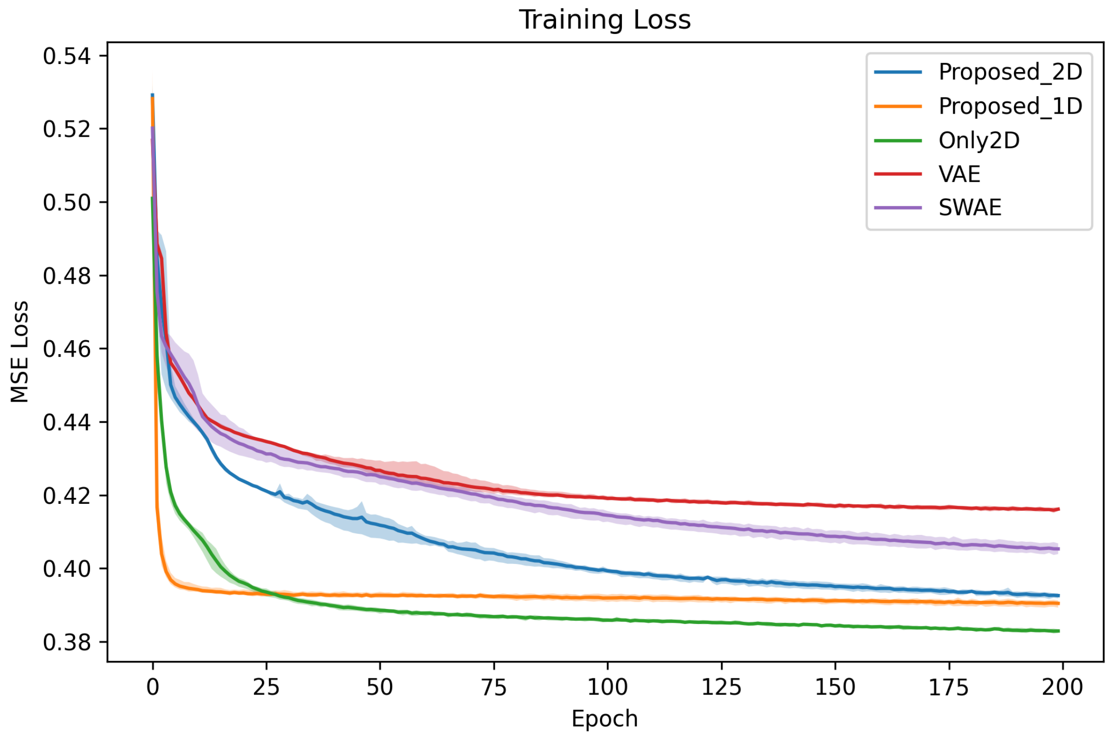


Figure 4.1.2: Model training loss per epoch trained. Averaged over 3 training runs each.

Our simple 2D auto-encoder forms the baseline for the the deep learning methods. As the baseline, it is the only method in this collection which relies only on a single optimization target. It focuses purely on the reconstruction loss with no additional loss terms to detract from this. This shows its benefits in the loss charts (4.1.2, 4.1.3). Here the simple auto-encoder showcases the fastest convergence to the lowest reconstruction loss. While this is not an altogether surprising finding, the question to keep in mind for later is how much the reconstruction loss alone plays a role in creating an appropriate latent representation.
Further, we move on to the first model using multiple optimization targets. The Sliced Wasserstein Auto-Encoder. This method is likely the most comparable to the basic 2D auto-encoder as it is exactly the same apart from the additional optimization target. This additional target concerns minimizing the Wasserstein distance between the observed distribution of the batch items in N dimensions (where N is the number of elements of the latent embedding) and a Gaussian centered around 0 in the same

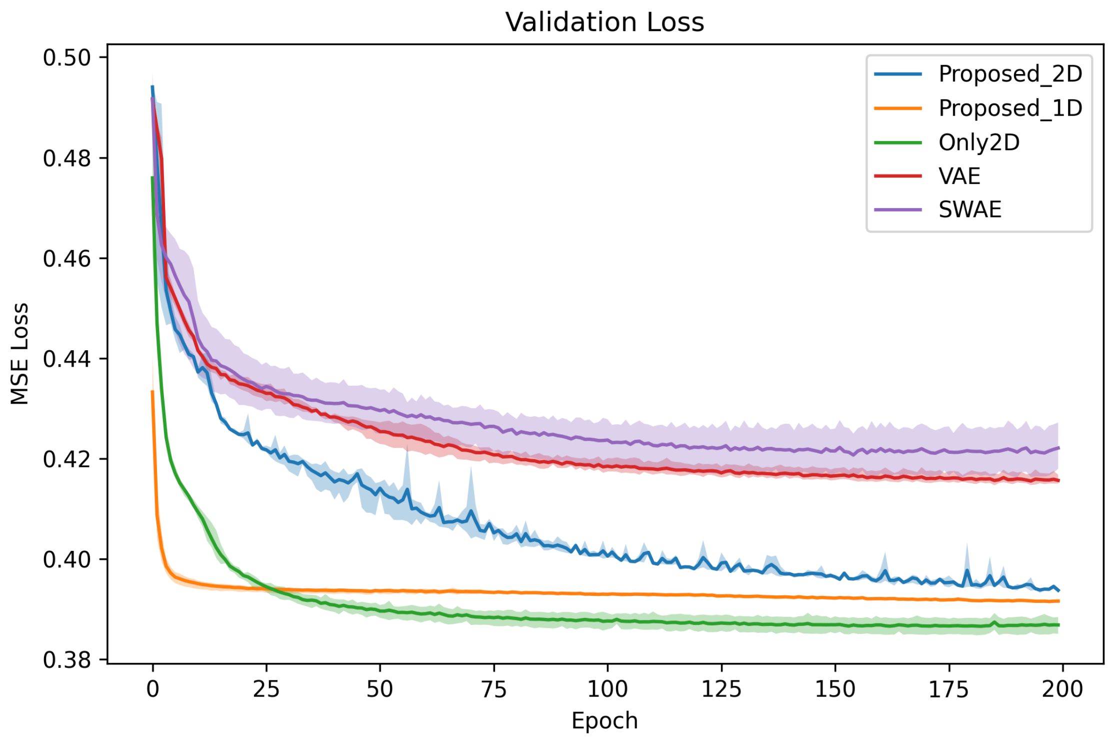


Figure 4.1.3: Model validation loss per epoch. Averaged over 3 training runs each.

space. The Wasserstein distance estimate quality is greatly dependent on the batch size, meaning that increasing it leads to more accurate results. After a coarse parameter search in this model space as well, the batch size was increased among all models to keep comparison stable. Generally a larger batch size (to a point) is beneficial even for other methods [71]. The additional optimization target, or the Wasserstein distance is given a weight here of just 0.01 in order to prevent it from interfering too strongly with the primary reconstruction target.
A method with even more difficultly tuning the weight of its secondary optimization target is the VAE model. Moreso than the SWAE model, the VAE model is extraordinarily sensitive to the λ scaling parameter. Set it too low and the network does not properly learn the distributions of the underlying data, set it too high and the MSE performance reaches a hard limit long before any of the other methods. In this case, the λ scaling parameter is set at 0.001, lower still than the SWAE extra target. Because of the reliance of this loss term on logarithmic operations, the scaling parameter is far more sensitive than the rather simpler to calculate Sliced Wasserstein Distance above. In this case, the scaling parameter was found to very strongly influence the minimum achievable reconstruction loss should it be balanced poorly.

4.1.3 Proposed Architecture

The training on the proposed architecture is done in two stages. First the 2D auto-encoder is trained with RICA regularization applied to the latent encodings for some level of sparsity. Later the two 1D auto-encoders are added to the end of this cascading system once the weights from the 2D auto-encoder are frozen. Training is then done for another 200 epochs on the same data in the same order. This separation in the cascading training can be seen in the loss charts as the separtated 2D and 1D

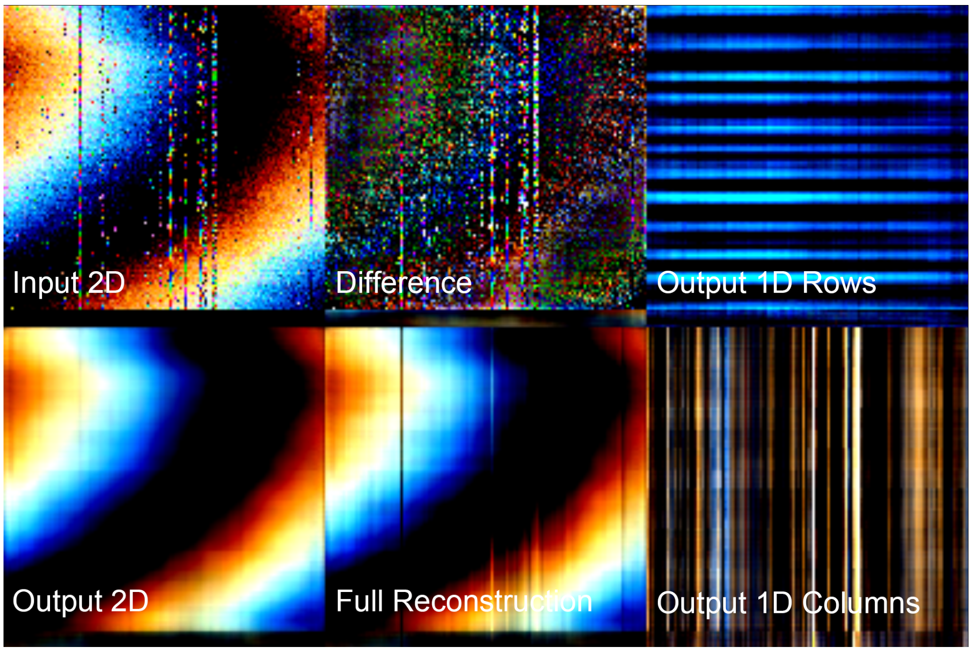


Figure 4.1.4: Showing how the output of the 2D and 1D auto-encoders used in our proposed cascading model differ and how they compliment each other when making the final reconstruction. Evident on the full reconstruction patch is that at least the vertical (frequency) components seem to have a marked effect in this particular example.

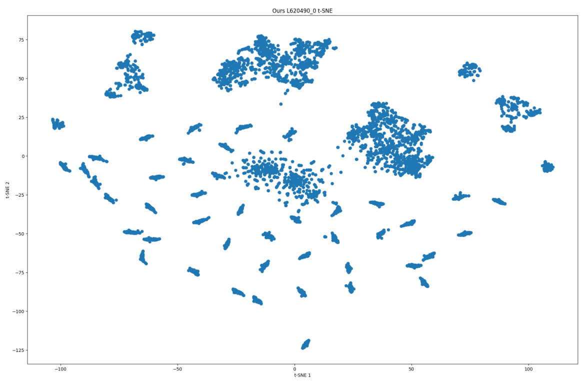
epoch loss plots. Results show only a marginal drop in MSE loss over this additional training. The plots show that this is perhaps one of the less stable methods with regards to validation and in some cases during training as well, the loss experiences sudden surges and dips. This may be caused by features changing to better fit the sparsity constraint then re-mapping further back with filters that have not changed much. Much like the KL-Divergence term for the VAE, here the RICA penalty is quite sensitive to changes in hyperparameters. While the scaling is not as big an issue, the learning rate applied most certainly is. With a learning rate of 0.001, the model will quite often fail to converge, meaning that this parameter has to be adjusted. It is in fact reduced by an order of magnitude to 0.0001 in order to support this additional loss parameter. By the looks of the loss plots for the 1D portion of the cascading encoders, this learning rate might still be somewhat too high, however, it does not seem to indicate any divergence.

4.2 Evaluating the Models

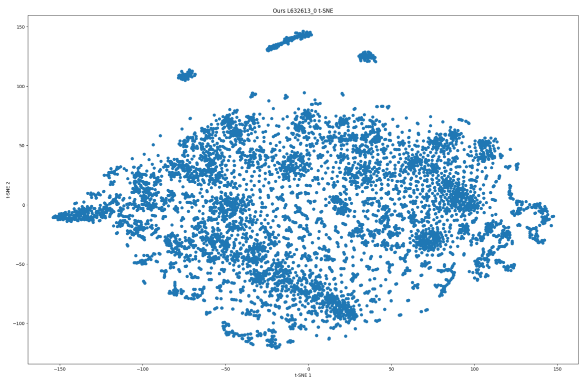
As described in the detailing of the labelling process in Section 3.3, a validation set of labelled examples is created using two observations; L620490 and L632613. Both observations appear in the HBA frequency region, however their distinction lies in the time scale at which the observation is made. For L620490 the original observation lasts 10 minutes and is downsampled to 128 time samples. By contrast L632613 is an observation spanning 8 hours which is downsampled to 121 samples in this reduced observation format. This presents a rather large difference in time scales between the two observations. In this case, we are not only comparing model performance between each model, but also accounting for how the model might be able to deal with different time scale features in the data. These time scales and more were all seen during the training process. The observations themselves were not seen during training or validation of the models, meaning that this data is entirely novel to

each. This means that the embeddings they generate should be entirely novel and a generalization of the previously learned data.
L620490 and L632613 are evaluated first separately then jointly. The joint embedding space is simply referred to as 'Aggregate' from this point forward for convenience. In essence the joint embedding is simply the concatenation of the embeddings a method generates for L620490 and L632613.
So far the models have been compared by their results in training, only insofar as things like reconstruction loss and other error terms may be compared. Because the training data does not include any labels, the following sections focus entirely on gauging each model's suitability for the given purpose through the embeddings that it generates of both the observations in the holdout validation set. These embeddings are considered either in their original dimensions, or when presented visually, through the two dimensional projection learned t-SNE. We move through the evaluations by first presenting the most qualitative evaluation method - visual inspection - then moving on through unsupervised evaluation towards the capstone quantitative classifier evaluation.

4.2.1 Visualizing the Latent Space



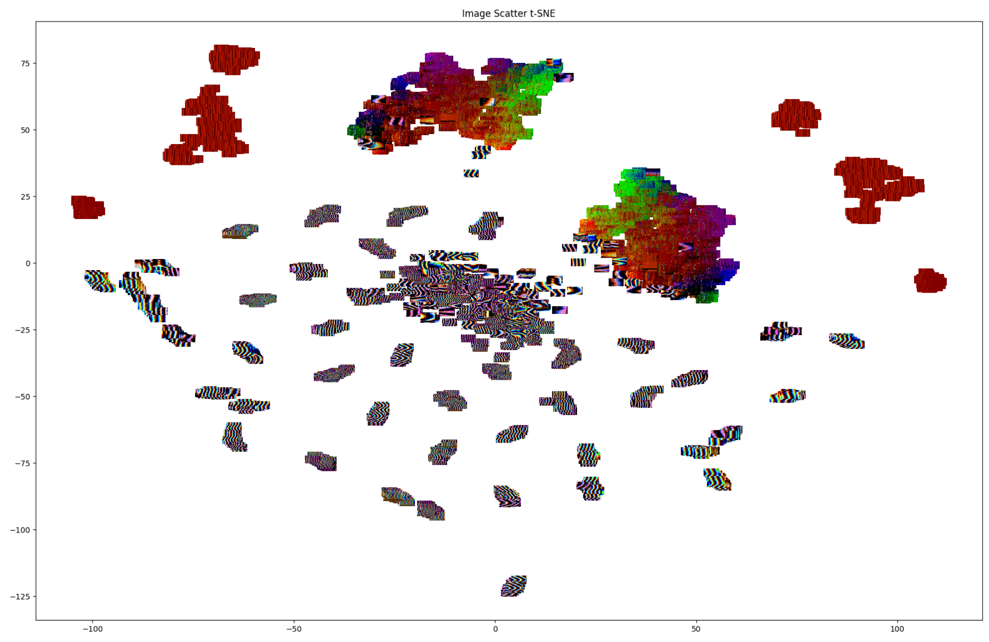
(a) Visualization of observation L620490 as captured by the 2D t-SNE embedding learned on the latent representation generated by our proposed method for each sample in the observation.



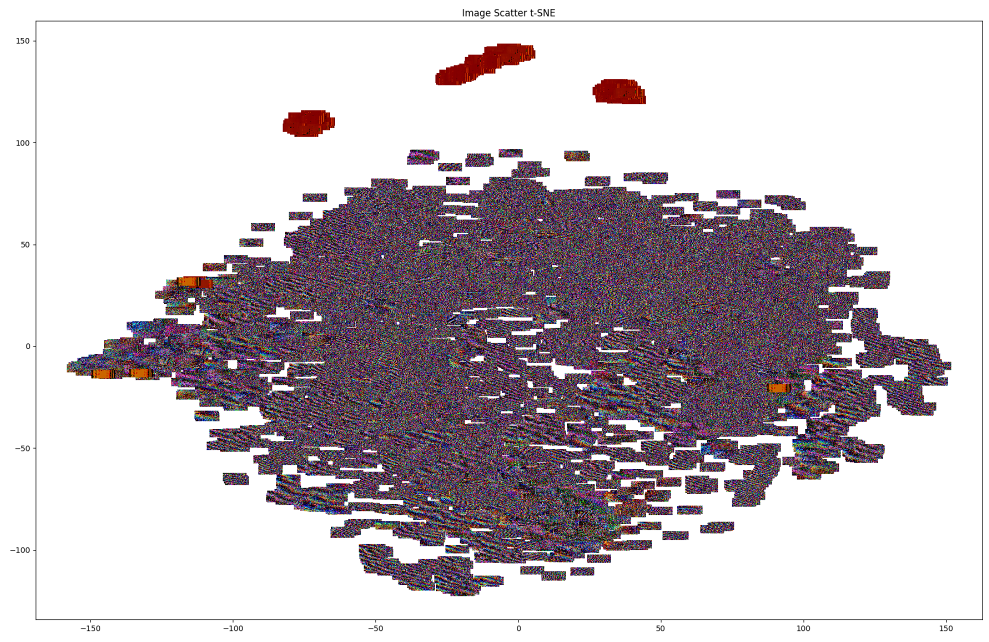
(b) Visualization of observation L632613 as captured by the 2D t-SNE embedding learned on the latent representation generated by our proposed method for each sample in the observation.

Figure 4.2.1: Showing t-SNE plots drawn from the 2D embedding learned from per-point latent embeddings generated by our proposed method for each of the two validation observations used in this study.

In order to better understand the data we are working with and the underlying characteristics of each of the two observations picked for the validation set, we first present a visualization of each. Both of the observations are shown in a plot which is obtained using a t-SNE embedding of the latent space generated by our proposed method. T-SNE as described in Section 2.3.3 is a manifold learning method which is useful for giving an approximation in lower dimension of the shape of data in higher dimension. In this case, our latent space consisting of 320 items (256 for the 2D portion of our cascading method then twice 32 for each of the 1D portions of the cascading design) down to a 2 dimensional representation. The plot for L620490 can be seen in Figure 4.2.1a and the corresponding plot for L632613 can be seen in Figure 4.2.1b. In addition to this, these same points were visualized with images of the corresponding baselines replacing the points in order to give a basic visualization of the groupings of visual features. These plots can be seen in 4.2.2a and 4.2.2b. From these four plots it becomes apparent that in observation L620490 there are several very visually distinct features



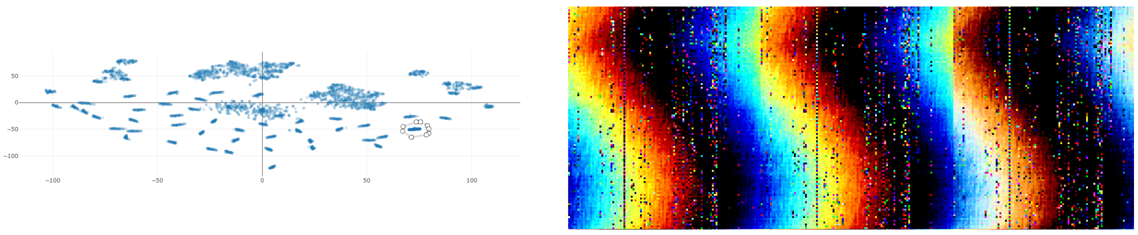
(a) Visualization of the latent space attached to the proposed architecture model for observation L620490. Note the large amount of very distinct small clusters which can be seen in this plot



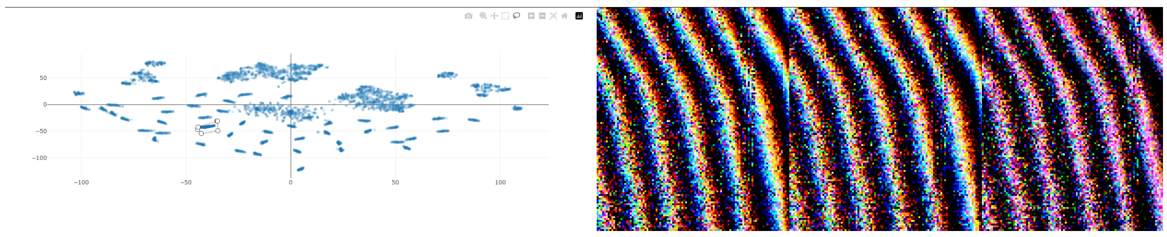
(b) Visualization of the latent space attached to the proposed architecture model for observation L632613. Aside from a few autocorrelations grouped together, the vast majority of the data appears to be grouped in a large indistinct clump.

Figure 4.2.2: Showing the same t-SNE plots as earlier in Figure 4.2.1, but this time the points are replaced by images of the corresponding baselines showing the correlation of visual and semantic similarity in the data.

which we can tell apart as well as several groupings of somewhat less clear cut visual features. For L632613, however there appears to be a much less clear cut latent space shape. Despite this, in the points example we can see some denser regions emerging in the embedding space. This becomes less apparent in the image plot where the images make a fine-grained view of the densities nearly impossible. Some features appear to stay distinct even in this observation. These samples appear to be autocorrelation baselines.
Already it is apparent from these visualizations that the longer time scale observations are not as feature rich. This may imply that many of the important features for separation lie in a somewhat shorter time-scale. While many similar features in L632613 still seem to group vaguely together, this effect is much more prominent in the case of L620490 where there appear to be extremely clear small groupings of very specific feature combinations. We make a further inspection of these regions using an application specifically developed for data inspection. In this application we show a bounding lasso around a selection of points in the t-SNE plot and the top three baseline spectrograms from that collection of selected points. Examples of this being used to inspect some of the smaller distinct clusters in the L620490 observation embedding space can be seen in Figure 4.2.6. The two distinct clusters selected in Figures 4.2.3a and 4.2.3b show quite clearly that the groupings in this region focus on the frequency of the overarching features. Inspecting other clusters shows that things like partial data loss (which was not caught in the overall labelling effort) show up in distinct clusters as well.



(a) Distinct cluster with very low-frequency features inspected in L620490 t-SNE embedding.



(b) Different distinct cluster with much higher frequency features inspected in the t-SNE embedding.

Figure 4.2.3: Showing the same t-SNE plot as shown above in Figure 4.2.1a side by side with a selection of points from there shown in full scale images for inspection. See selection box on the left, images shown on the right.

Examining also the latent space drawn from observation L632613 in the same way, we can see that similar regions do indeed group similar features, but it is hard to pick up boundaries between various conditions. Figure 4.2.4 shows one selection of a dense region in L632613. Selecting larger areas around these denser points however, shows a much more chaotic picture indicating again that the boundaries in these areas are not very strictly defined at all. The baselines corresponding to autocorrelation seem to be the most similar to how they appear in L620490 in that they are the only ones distinctly sequestered from the general data population.
Overall the closer manual inspection of all the models latent spaces reveals similar findings for both

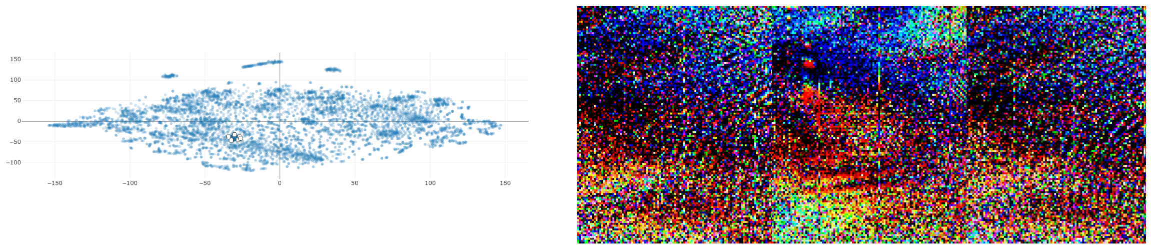


Figure 4.2.4: Dense region of L632613 inspected. Showing fewer signs of similarity between the top 3 results than L620490 earlier.

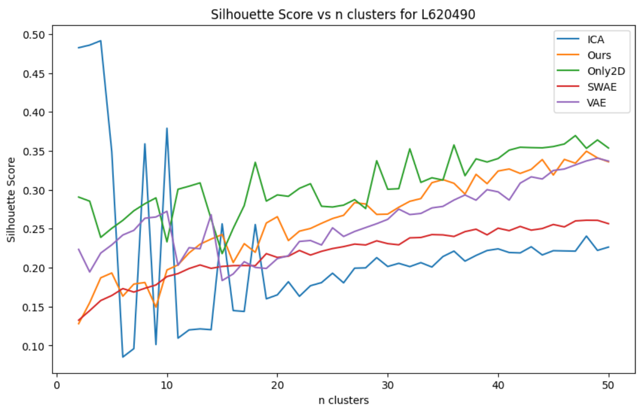
observations. L620490 groups certain items into a few large clusters of similar data and many smaller outlying clusters which show very distinctive features between each cluster. By contrast, observation L632613 seems to always form less distinct boundaries between various feature groups. Subjectively speaking it appears that the models with the best reconstruction loss performance make for the best embeddings, at least insofar as this can be judged when the original data sits behind 2 layers of abstraction (namely an auto-encoder, then t-SNE).

4.2.2 Unsupervised Clustering Evaluation

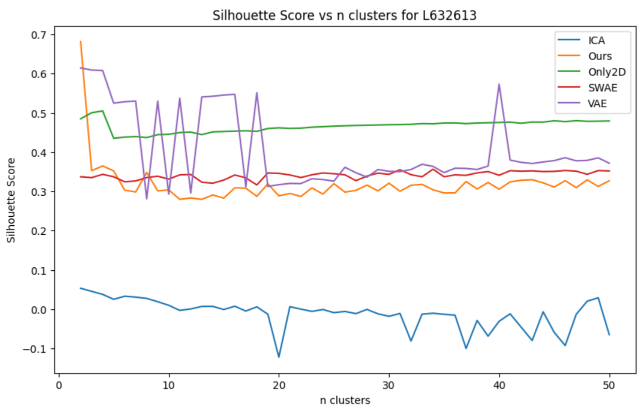
To begin with, we treat the found latent space as a data mining problem. For this, we apply two clustering methods through which we attempt to find the optimal partitioning of the latent space in order to best describe the underlying data. This is done in an attempt to find a number of relevant features which could then be exploited in the further description of the data. Applied are k-means clustering and Gaussian mixture modelling, then on each of the found clustering partitions defined by the fit models, we apply either the Silhouette Statistic or the Bayesian Information Criterion (BIC). In each of the conditions, the true clustering solution is assumed to be somewhere between 2 and 50 distinct clusters, meaning that this is the range for which we test each of the methods below. Testing outside of this range becomes extremely slow and unwieldy for the Gaussian mixture models, making this the primary restriction on the numbers of clusters tested.

k-means clustering

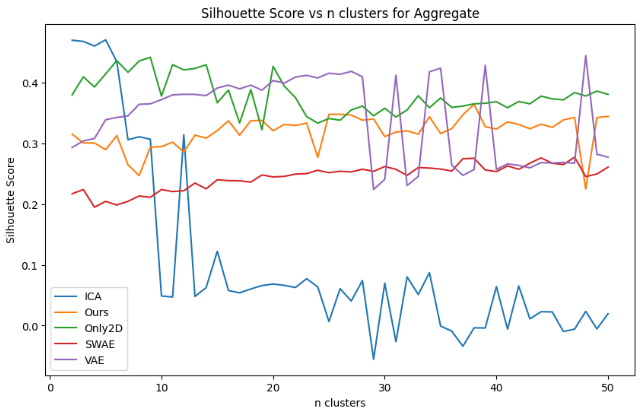
As a basis for the unsupervised evaluation of the latent embeddings, we apply the fastest and most basic clustering algorithm, namely k-means clustering. To evaluate the quality of the clustering we use the mean silhouette score overall for the entire dataset. For a few selected models, we additionally show the per-point scores in order to show more detail. This aids in showing which clusters in either observation are most distinct.
Average silhouette scores can be seen per observation condition in Figure 4.2.5. From this plot it can easily be seen that there is no very distinct winner for the number of clusters in an arrangement for any of the three observation conditions. In the case of the L620490 observation there appears to be a section around 18 clusters where both ICA and the basic 2D auto-encoder agree on the optimal number of clusters. After this point the near monotonic increase in the silhouette score indicates it is likely that only the model complexity further contributes to increasing the score. In the plots shown in Figure 4.2.5 higher score indicates better performance.
In data for observation L620490 the performance of the auto-encoder models seems to follow roughly their earlier reconstruction losses where better reconstruction loss seems to lead to better groupings under k-means. ICA briefly tops the charts, then becomes erratic and unstable before settling at



(a) Silhouette scores per condition for L620490



(b) Silhouette scores per condition for L632613



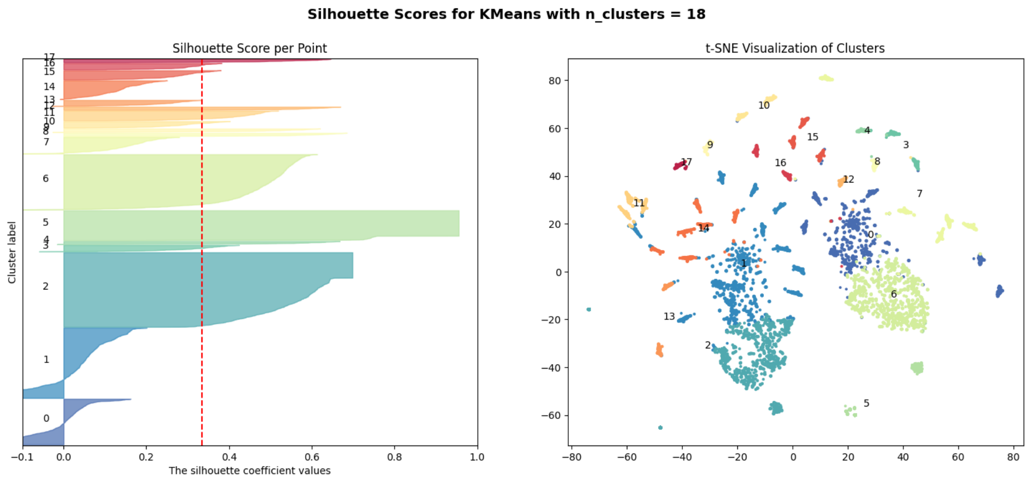
(c) Silhouette scores per condition for Aggregate

Figure 4.2.5: Presenting the silhouette scores based on the k-means clustering partitions for each of the three conditions in the dataset. Higher is better. The plots show very high average scores at lower *k*. Some models, such as ICA and VAE also show quite high levels of instability over the range of data conditions.

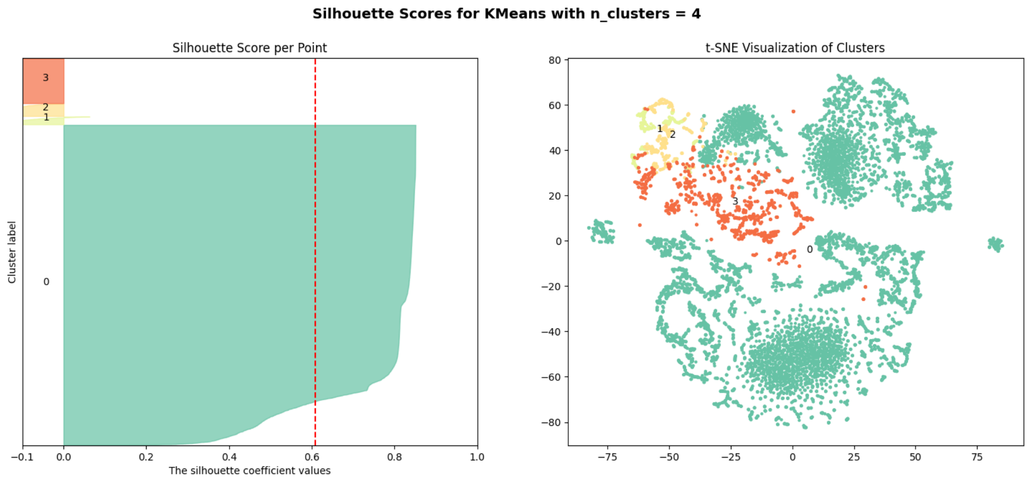
the lowest average score. Notably for L632613, ICA performs markedly worse than the rest of the methods and this translates into the joint embedding space as well in the aggregate condition. The other notable oddity in the chart is the high degree of instability which the VAE model experiences in the L632613 observation's embedding. While this instability carries over to the aggregate condition it appears at a higher *k* possibly giving away at least information about the relative quality of the two sets of data.

For further investigation we pick out a few of the models and analyze their silhouette scores on a per-point basis. To do this we pick out the silhouette plots of the basic 2D auto-encoder with 18 clusters for the L620490 observation, the silhouette plot with 4 clusters for the VAE case in the L632613 observation and lastly the silhouette plot of our proposed architecture with 26 clusters for the aggregate observation case. These plots are shown in Figures 4.2.6a 4.2.6b and 4.2.6c. This is done because the average silhouette scores presented in the previous plots show only part of the picture, the overall score. However we can use the per-point silhouette scores to examine each of the latent spaces and see if there are clusters which are nevertheless solidly grounded and separated from the rest of the data despite the average score over the dataset not indicating anything of value.

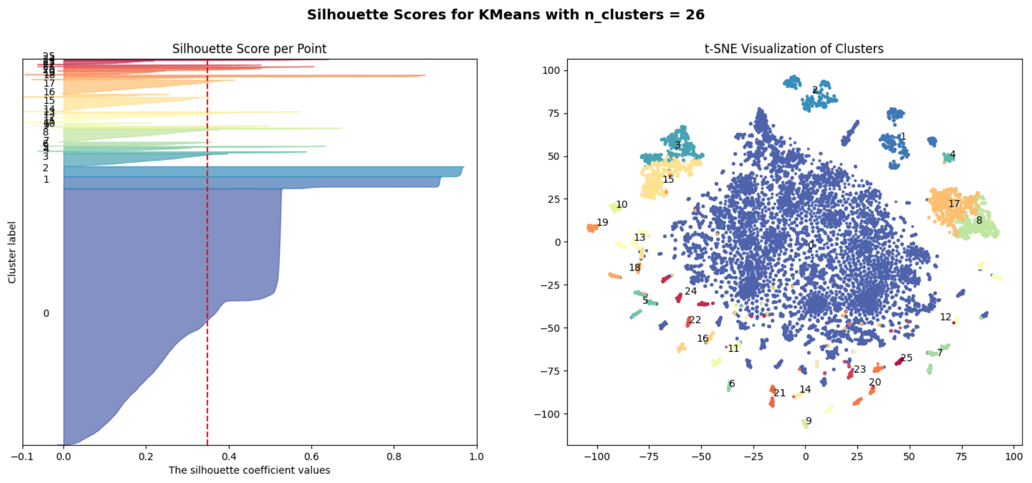
The silhouette scores in these trials show rather different things for the different observations. First, observation L620490 shows that the small distinct clusters of features are being identified, however



(a) Per-point silhouette score with cluster membership for $k = 18$ and t-SNE embedding of L620490 using Only2D model latent space.



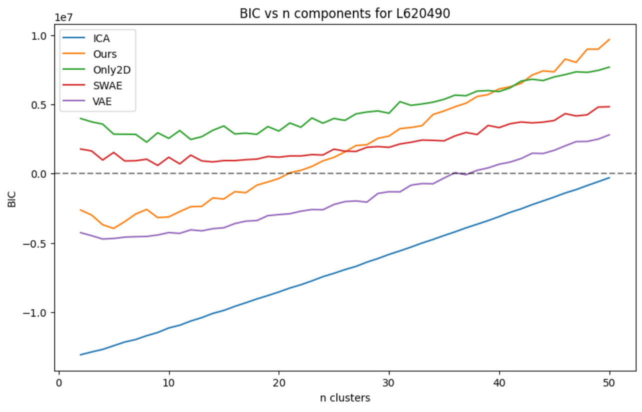
(b) Per-point silhouette score with cluster membership for $k = 4$ and t-SNE embedding of L632613 using VAE model latent space.



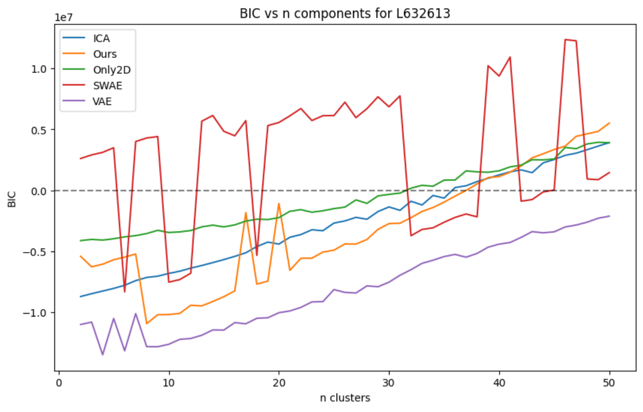
(c) Per-point silhouette scores with cluster membership for $k = 26$ and t-SNE embedding of the aggregate observations using our proposed model's generated latent space. The large dark blue blob notably contains the entirety of L632613 separated from L620490.

Figure 4.2.6: Selected silhouette scores for clustering conditions covering each of the three observations, the two original separate ones and the aggregate set. Red lines in the plots indicate the average silhouette score which is then reported also in the Figure 4.2.5 above.

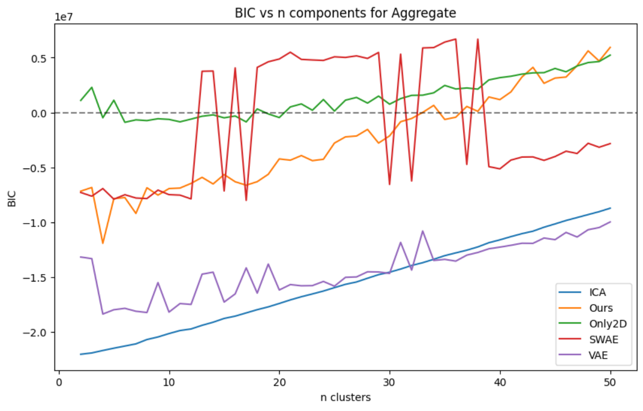
in batches rather than individually. It is also rather evident that there are two somewhat troublesome clusters of data which are indistinct in this case as well. They can be seen in Figure 4.2.6a as clusters 0 and 1 with near zero or even negative silhouette scores for member points, the most distinctly egregious clusters of this lot. Second, in Figure 4.2.6b we can see that where observation L632613 is concerned the results become rather strange. In the case of the VAE model, we find in the latent space that 4 clusters seems to fit best according to the silhouette score, however this manifests as simple grouping of almost all the data in a single large cluster and the exclusion of some smaller clusters all of which appear to be very poorly defined. Of course, here is where some of the pitfalls of t-SNE show through, where it is a generally reliable method for learning a projection by which to represent the data in lower dimensional space, however here the shape of the clusters does not seem to match what the silhouette score shows very closely. Lastly the aggregate case seems to de-mystify this somewhat by showing that largely observation L632613 forms an entirely distinct cluster when joined with the rest of the data. Then the smaller clusters of distinct features once again spring up in the surrounding area. Finally this shows that picking the highest silhouette score is not always favourable, which makes this a somewhat unreliable evaluation method.



(d) BIC scores per condition for L620490



(e) BIC scores per condition for L632613



(f) BIC scores per condition for Aggregate

Figure 4.2.6: Presenting the Bayesian Information Criterion scores based on the Gaussian mixture modelling partitions for each of the three conditions in the dataset. Lower is better. Of particular note are the very stable scores for L620490 and the extreme instability of SWAE in both the L632613 and Aggregate conditions.

Gaussian Mixture Modelling

With a latent size of 320 for our proposed architecture and 256 for the rest of the methods, we find ourselves right at the limit of what Guassian Mixtures can handle (See 2.6.2 for clarification). Compared to the faster k-means, this method may be able to tell us more about the underlying data.
Following the model fitting procedure, we use this to compute the partitioning scheme by assigning each point to its most likely parent distribution. This partitioning scheme is then used to determine the BIC score for a particular set of components. BIC adds a penalty term for model complexity (number of components) meaning that the lower the score is the better.
Results of this evaluation can be seen in Figure 4.2.6. In the lower time-scale observation (L620490) While Gaussian mixtures are generally a more robust clustering method than k-means given its greater flexibility to find odd shapes in the data, it is far less scalable and almost not appropriate at all in the case of the data which we find ourselves dealing with here. The 256 dimensional data and especially the 320 dimensional data associated with our proposed architecture is reaching the limit of what can and should be processed with Gaussian Mixture Modelling. Refer back to Section 2.6.2 for clarification on how data dimensionality impacts the computational complexity of a model fit in a cubic relationship. Components are not equivalent to clusters in k-means, however in the case of this problem where we are trying to find overlapping attributes in the data space, Gaussian Mixtures may prove to be more useful in determining points which have a chance of being drawn from more than one distribution. To evaluate this method using the Bayesian Information Criterion, we still make a hard partition where we assign points to the distribution it is most likely to have been drawn from, however. In an inversion of the previous plots associated with k-means and the silhouette statistic, here the BIC score indicates a better model when it is lower rather than higher. This is because BIC penalizes the model quite heavily on increased complexity as can be seen by the monotonic increase of the score as the number of components grows. Starkly contrasting with the previous method, ICA in this case on the L620490 observation appears as a linear plot with its lowest point being the 2 component condition, then continuing with nearly no variation until coming to rest near 0 at 50 components. Either this is an indication that the solution should be given by one single component here or this statistic is not well applied to this problem. The proposed architecture stands out here for one particular reason, this being that with increased model complexity the degradation in BIC score happens the fastest. Whereas the BIC scores initially are comparable with the variational autoencoder architecture, they are barely comparable with the simple 2D architecture here. Because of how heavily the model complexity is penalized in the BIC statistic however, this different rate of increase may be adequately explained by the additional 64 dimensions in the proposed architecture's latent space.

4.2.3 Classifier Evaluation

In addition to the unsupervised evaluation, the labels as defined in Section 3.3 are leveraged for supervised evaluation. The simple classifiers are trained on the labels and the embedding outputs of each of the models. For the classification, we select the Random Forest model and a Gaussian Naive Bayes models as our classifiers. As explained previously, the labels are multi-attribute, meaning that each point may have one or more label classes assigned to it. These classifiers are chosen because they are both able to deal with multi-attribute data. In both cases this means that in the background the classifier is configured as a 'one vs rest' discriminator per class. The evaluation is done by splitting the labels and data up using a stratified split which is necessary because of the enormous label class imbalance. Once again see Section 3.3 for details. In the end there are sufficient samples to do a 5-fold cross validation the results of which are reported in Tables 4.1, 4.3 and 4.4.

Table 4.1: Classifier performance in % accuracy

Embedding	Random Forest	Naive Bayes
	L620490	
ICA	91.17 ± 1.1	35.23 ± 2.0
Only2D	92.96 ± 0.3	44.26 ± 0.8
SWAE	90.95 ± 0.9	**67.08 ± 0.4**
VAE	**93.26 ± 0.4**	43.86 ± 3.1
Ours	92.89 ± 0.3	59.39 ± 1.5
	L632613	
ICA	40.21 ± 1.1	28.16 ± 1.9
Only2D	48.55 ± 0.8	22.82 ± 7.6
SWAE	42.24 ± 0.4	21.19 ± 1.6
VAE	**49.51 ± 0.4**	14.62 ± 0.9
Ours	47.57 ± 1.1	**40.09 ± 4.9**
	Aggregate	
ICA	**66.3 ± 0.8**	30.44 ± 1.2
Only2D	64.62 ± 0.6	17.43 ± 0.8
SWAE	60.2 ± 0.6	**42.21 ± 0.4**
VAE	64.59 ± 0.5	11.69 ± 0.2
Ours	65.03 ± 0.7	27.97 ± 0.4

Table 4.1 demonstrates the performance of the classifiers across different embeddings. The Random Forest classifier's accuracy ranged between 40.21±1.1 (ICA for L632613) to 93.26±0.4 (VAE for L620490), while the Naive Bayes classifier exhibited a wider range of 14.62±0.9 (VAE for L632613) to 67.08±0.4 (SWAE for L620490). The aggregate accuracy shows ICA leading with 66.3±0.8 in Random Forest and SWAE leading with 42.21±0.4 in Naive Bayes. Our proposed method performs above average in the L620490 condition under GNB (59.39±1.5) and beats other methods in L632613 with the same classifier by a comfortable margin (40.09±4.9) performing less favourably in the joint observation condition. Its performance in the RF classifier condition is consistently above average, however never reaching the top.

Table 4.2: Total averages over all classifier and observation performance conditions.

Model	Total Average
ICA	48.59 ± 1.4
Only2D	48.44 ± 1.8
SWAE	53.98 ± 0.7
VAE	46.26 ± 0.9
Ours	**55.49 ± 1.5**

Considering the average performances over all data and classifier conditions, we arrive at the figures presented in Table 4.2. Here we show that overall our method is the most well rounded, though not

by a significant margin.
From the tables, it is evident that the Random Forest classifier consistently outperforms the Gaussian Naive Bayes classifier. For instance, in the Aggregate section, the Random Forest classifier's lowest accuracy is 60.2±0.6 (SWAE) compared to Naive Bayes' 11.69±0.2 (VAE). This performance difference is likely due to the Naive Bayes' assumption of independence in features, a condition not met in all models.
The Sliced Wasserstein Auto-Encoder (SWAE) appears to be the most compatible with the Bayesian classifier, with a remarkable performance of 67.08±0.4 in L620490. This is closely followed by the proposed method (59.39±1.5), though it falls behind in the aggregate case (27.97±0.4), still outperforming the basic 2D auto-encoder (17.43±0.8) and VAE (11.69±0.2). The SWAE's enforcement of an N-variate Gaussian distribution on the latent space seems to synergize well with the Naive Bayes' assumption of Gaussian feature distribution.

Table 4.3: Table showing the F1 Scores for the Random Forest classifier for each algorithm and condition. Where a user assigned label is not present, no observations containing it are present at the time.

Label	Instances	ICA	Only2D	SWAE	VAE	Ours	Average
			L620490				
Autocorrelation	132	0.13	0.91	0.37	0.91	0.89	0.64
Data Loss	678	0.96	1.00	1.00	1.00	1.00	0.99
Decorrelation	344	0.41	0.61	0.40	0.64	0.63	0.54
Source Structure	2438	0.96	1.00	0.98	1.00	1.00	0.99
Vert. Artifacting	70	0.57	0.28	0.67	0.42	0.25	0.33
			L632613				
RFI High	2300	0.76	0.77	0.74	0.76	0.78	0.76
Autocorrelation	124	0.91	0.99	0.89	0.93	0.97	0.94
CB Artifacting	1130	0.09	0.39	0.29	0.48	0.38	0.33
Source Structure	3780	1.00	1.00	1.00	1.00	1.00	1.00
			Aggregate				
RFI High	2318	0.65	0.74	0.59	0.72	0.71	0.68
Autocorrelation	256	0.8	0.94	0.84	0.92	0.93	0.89
CB Artifacting	1130	0.01	0.31	0.29	0.37	0.35	0.27
Data Loss	678	1.00	1.00	1.00	1.00	1.00	1.00
Decorrelation	346	0.59	0.64	0.30	0.62	0.61	0.55
Source Structure	6218	0.98	1.00	0.99	1.00	1.00	0.99
Vert. Artifacting	70	0.20	0.03	0.00	0.03	0.17	0.09

The F1 scores further support these findings. Table 4.3 shows high consistency in Random Forest, with the Source Structure label maintaining a 0.99 average across different observation conditions. The Naive Bayes classifier, however, shows more variability as seen in Table 4.4, where the scores for Autocorrelation fluctuate from 0.13 (ICA) to 0.38 (Ours) in L620490.
In general the F1 scores show a more granular overview of the classifier performance. This also gives an insight into how well label classes are defined within the dataset. As already mentioned, the

Table 4.4: Table showing the F1 scores reported per label class for each of the conditions using a Naive Bayes classifier. Where no label is assigned, no item with that label appears in the set considered. Note that the ’vertical’ and ’checkerboard’ artifacts are exclusive to the long time-span observation L632713

Label	Instances	ICA	Only2D	SWAE	VAE	Ours	Average
			L620490				
Autocorrelation	132	0.13	0.18	0.37	0.14	0.38	0.24
Data Loss	678	0.96	1.00	1.00	1.00	1.00	0.99
Decorrelation	344	0.41	0.34	0.4	0.35	0.32	0.36
Source Structure	2438	0.96	0.95	0.98	0.96	0.95	0.96
Vert. Artifacting	70	0.57	0.68	0.67	0.69	0.50	0.62
			L632613				
RFI High	2300	0.61	0.71	0.6	0.43	0.51	0.57
Autocorrelation	124	0.96	0.21	0.24	0.08	0.83	0.46
CB Artifacting	1130	0.47	0.35	0.39	0.42	0.41	0.41
Source Structure	3780	1.00	0.93	0.94	0.45	0.99	0.86
			Aggregate				
RFI High	2318	0.73	0.66	0.55	0.54	0.65	0.63
Autocorrelation	256	0.21	0.27	0.46	0.09	0.63	0.33
CB Artifacting	1130	0.45	0.4	0.38	0.31	0.4	0.39
Data Loss	678	0.92	1.00	1.00	1.00	1.00	0.98
Decorrelation	346	0.44	0.21	0.37	0.23	0.23	0.30
Source Structure	6218	0.98	0.97	0.97	0.59	0.99	0.90
Vert. Artifacting	70	0.04	0.03	0.29	0.04	0.03	0.09

Source Structure label class is very well defined throughout, or simply widely enough applicable with enough examples available to be easily classifiable. Both classifiers and all methods show a very high F1 score for this class label, averaging at 0.99. Data loss is another well defined class of which fewer examples are available. Looking back to the data inspection portion of the evaluation, data loss is visible in its own compact and distinct grouping in Figure 4.2.2a.
Decorrelation seems very well defined according to the RF classifier with an average F1 score of 0.89 where this score is 0.36 for the GNB classifier. Similarly the Autocorrelation condition appears to depend heavily on the method applied regarding its exact performance. Notably ICA and VAE perform consistently poorly in this category.

Chapter 5

Discussion

This thesis proposes a novel cascading auto-encoder to deal specifically with oddities in spectral LOFAR data. The adaptation focuses on picking up high frequency features in both the time and frequency domains separately. Finally with the results presented in the previous chapter, we have the means to give an answer to the question posed at the start of this work. Namely, what is the best visual representation learning method to apply to spectral data in order to facilitate downstream tasks such as data inspection, label propagation, anomaly detection and classification? This section first goes through a brief interpretation of the results seen in the previous chapter, focusing most strongly on the classifier results. This analysis is used to answer the research questions posed in the introduction. Following this, a few discussion points are raised about the methodology comparing it to related work. Lastly we present a critique of of the applied methodologies and give pointers for further research.

5.1 Visual Inspection

Visually inspecting the data is certainly an unusual way to start model evaluation. Given that part of the task of this project concerns data inspection as well as potential for further labelling using such visual means it is salient to explore. As shown in Figures 4.2.2a and 4.2.2b, our proposed method shows some promise regarding data inspection tasks. Specifically with the L620490 observation, the abundance of distinct cluster groupings as further elaborated on in Figure 4.2.6 makes clear that we have a good starting point for building this zoo of artifacts. Which may combine user expertise and simple visual features to create a more extensive labelled dataset for further study. In Figure 4.2.4 we show that this does not hold quite as robustly for longer time-scale observations such as L632613. This does not preclude such methods from being applied anyways with some more manual effort involved.

5.2 Clustering Validity Indices

In the clustering evaluation, we see that while the visual inspection and t-SNE appear to very strongly map certain features and groups together, clustering appears to struggle with creating sensible partitions of the subspaces. In addition to gaining some more insight into the data as well as the problem of the curse of dimensionality, we may glean some relative model performance from this analysis. In particular, ICA and VAE appear to experience some stability issues in the silhouette score case. Something similar occurs with SWAE when performing the BIC analysis on the Gaussian mixture models. Showing extreme levels of instability where other methods experience only minor fluctuations. Neither of these analyses show any conclusive evidence for the correct number of clusters or components being in the range of 2 to 50. Overall this points to difficulties using such methods to perform unsupervised analysis on the LOFAR dataset embedded in this way. Specifically certain anomaly detection methods relying on unsupervised methods may struggle to accurately identify the relevant data, for example.

5.3 Classification

Evaluation using the GNB and RF classifiers represents the core of the quantitative analysis performed in this thesis work. The classification performance is shown in the results over five folds of cross validation with stratified data splitting to account for the highly unbalanced classes. This provides enough data to be able to say something concrete about both the data and the methods employed.
The overall poorer performance of all the classification tasks on the L632613 observation shows that the longer time scale observation with more data lost in the downsampling procedure no longer retains good enough feature density to make for good embeddings. See Table 4.1 for the general classifier performance comparison. Some confirmation at least for the separate nature of the two observations in the latent space is shown by 3 of the 5 methods attaining the best silhouette score in the k-means clustering task at just 2 clusters. While the joint embedding still shows solid performance, it is clear from Tables 4.3 and 4.4 that the performance is largely held up by the rather ubiquitous *source structure* class which appears well delineated in both sets of data.
Comparing the performance of the RF and GNB classifiers against each other with the same models on the same dataset, we observe some differences. GNB performs markedly worse in all but two cases. This can be seen by considering the averages per label class in Tables 4.3 and 4.4, as well as per model condition in Table 4.1. This is likely due to the more stringent assumptions that GNB makes about the underlying data [67] which RF does not make. The other notable difference here may be the greater number of parameters and flexibility of the RF model.
ICA's performance in the classification tasks is middling overall, though it does outperform all other methods in the aggregate data condition using the RF classifier. It is somewhat surprising to see the only method which does not rely on deep learning achieve this score. In the aggregate condition, even its average score across both classifiers is impressive showing that, despite relatively poor performance on L632613 and middle of the pack performance on L620490, it is somehow able to cope well with the combination of both. The F1 scores however, show the method struggling quite a bit to cope with certain rather distinct classes such as autocorrelations.
Our baseline 2D auto-encoder model performs remarkably well through the categories, showing that solid reconstruction convergence goes a very long way to solving the classification problem. Despite this, it does not clearly take the lead in any of the dataset or classifier conditions. This reinforces the idea that the regularization and choice of secondary target are equally important in this problem space.
Because of GNB's assumption that the features follow a Gaussian distribution, the subspaces created by SWAE appear to fit it best in 2 of the 3 dataset conditions. Surprisingly, with L632613, our method outperforms SWAE. This is where SWAE has what appears to be uncharacteristically low performance. This is corroborated by the strange instabilities in the BIC plot for the same observation condition. It is unclear if this is due to the combination of SWAE's least ideal reconstruction loss convergence point and observation L632613's over-compressed features or because of its regularization conditions.
Largely our method performs above average in most label categories only dipping below the method average score in the *vertical artifacting* class in both GNB and RF cases. Using the GNB classifier, we note this below-average performance also in over half the label classes in L620490, the *RFI high*

condition in L632613. The poor comparative performance with regards to the vertical artifacting label class is rather surprising given that Figure 4.1.4 shows specifically an improvement in the reconstructioni ability of vertical and horizontal elements after considering the base 2D reconstruction. We noted that the base 2D auto-encoder's performance comparatively mirrored its reconstruction loss scores. We had hoped that this enhanced ability to reconstruct specific features would give our method an advantage here.

5.4 Evaluating Representation Learning Methods for Spectral LOFAR Data: Which Yields the Optimal Embeddings?

In our analysis of six different classification tasks, the proposed cascading auto-encoder method achieved top classification accuracy only once, but it outperformed all other methods in overall average performance. This advantage, however, comes with a drawback—it is twice as computationally intensive to train compared to other models, even more when accounting for the additional inference that three separate auto-encoders leverage. Although this is somewhat offset by the sparsity constraint, the method's reliance on the largest latent space representation indicates that it may not be the ideal choice after all.
Variational Autoencoder (VAE) fares poorly overall due to its weak performance with the Gaussian Naive Bayes (GNB) classifier but performs best under the Random Forest condition. The SWAE model aligns well with the Naive Bayes classifier, as GNB's assumptions align closely with the distribution that SWAE imposes on the latent dimension. Both VAE and SWAE exhibit some instability during clustering tasks, but this is not a significant concern since clustering evaluation is less precise than quantitative evaluation.
The baseline 2D auto-encoder illustrates the importance of latent constraints over achieving the best absolute reconstruction loss, in achieving an optimal encoding. Independent Component Analysis (ICA) deserves mention for its explainability, though it is also the least scalable and extendable. In fact, our research pushed the method nearly to its limit where computational feasibility is concerned.
The final model selection depends on the specific goals. For a generative model, VAE is optimal; for an easy-to-train model with softly enforced latent space boundaries, SWAE is preferred. If the aim is to train a model on more data while maintaining a reasonably sized latent space, our proposed method suggests further learning, at least in the 2D layers, may be possible for the cascading approach.
Addressing the second part of our research question, the somewhat inconclusive results from clustering evaluations highlight the importance of proper application of these methods for understanding data structure. To make substantial progress in evaluating self-supervised methods, a set of labeled data is essential.

5.5 Comparison to Related Work

Previously work has been done on the problem of System Health Management with regards to LOFAR data [18]. This method applies a VAE which, by contrast to our work, combines the use of both phase and amplitude data obtained from observations. The method is trained on artificial data for evaluation as opposed to creating a seed label set for future work. Furthermore, the data pre-processing, rather than applying a pattern recognition approach (where absolute data positions are not preserved), the authors apply interpolation to make the data fit into a fixed input size model. Overall, the end goals of the two methodologies are aligned, we simply show that a different approach is also viable. The lack of a pre-existing labelled dataset makes comparative evaluation difficult, however it is hoped that this work may push the envelope on that particular hurdle.

5.6 Limitations and Future Work

This section aims to outline the key challenges faced during this research and propose directions for subsequent investigations. Being the second phase of an ongoing research endeavor, this study has undoubtedly contributed to the field by introducing tools and techniques that facilitate faster and more effective sample labeling. Nevertheless, this also leads us to the first significant limitation.
The scarcity of labels available for evaluating this research proved to be an obstacle. Though two observations worth of labeled data allowed for some conclusions about the problem's nature and the models used, a larger quantity of labels would have undoubtedly led to more robust and conclusive results. A feasible solution might include utilizing the data inspection tools developed in this work to label large data portions rapidly. Achieving a balanced dataset may still remain a challenge due to inherent distribution attributes, but the introduction of anomaly detection elements in this thesis work anticipates issues like disbalanced classification and continuous system operation.
Regarding the clustering evaluation, a reconsideration of approaches may have been beneficial. Specifically, discarding the poorly scaling Gaussian mixture modeling in favor of expanding the clustering search with k-means clustering could be explored. However, high-dimensional spaces present the curse of dimensionality, where traditional metrics such as Euclidean distance lose efficacy. Future work may include exploring alternative distance measures, even though such efforts were not fruitful in this research.
Expanding the training dataset further is another option. Given that the pattern recognition approach seems to work rather well, the minimum data size requirement may be reduced and smaller patches utilized instead in order to expand the pool of available data. This work explored self supervised learning in broad strokes, applying several different and varied methods, future work may consider exploring methods more in depth. Stronger sparsity constraints with appropriately picked pretext tasks may yet yield better results. Rotation and jitter as methods of data augmentation may not be ideal given the problem domain, however perhaps a contrastive loss method further exploring specifically image plane coordinates may indeed be appropriate.
This thesis focuses strongly on visual methods and adapting them to the domain of spectral data. A comparison to similarly sized sequence learning methods is a salient future direction as well. From basic RNNs and LSTMs to more modern, Transformer-based approaches.

Bibliography

[1] Donghwoon Kwon et al. “A survey of deep learning-based network anomaly detection”. In: *Cluster Computing* 22.1 (2019), pp. 949–961.

[2] Maarten Meire and Peter Karsmakers. “Comparison of deep autoencoder architectures for real-time acoustic based anomaly detection in assets”. In: *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 2. IEEE. 2019, pp. 786–790.

[3] Menno Liefstingh et al. “Interpretation of Deep Learning Models in Bearing Fault Diagnosis”. In: *Annual Conference of the PHM Society*. Vol. 13. 1. 2021.

[4] Charu C Aggarwal. “An introduction to outlier analysis”. In: *Outlier analysis*. Springer, 2017, pp. 1–34.

[5] “Novelty Detection”. In: *Network Security* 2003.3 (2003). 18, pp. 18–19. ISSN: 1353-4858. DOI: 10.1016/S1353-4858(03)00313-1. URL: https://doi.org/10.1016/S1353-4858(03)00313-1.

[6] Markos Markou and Sameer Singh. “Novelty detection: a review—part 2:: neural network based approaches”. In: *Signal Processing* 83.12 (2003), pp. 2499–2521. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2003.07.019. URL: http://www.sciencedirect.com/science/article/pii/S0165168403002032.

[7] Elaine R. Faria et al. “Novelty detection in data streams”. In: *Artificial Intelligence Review : An International Science and Engineering Journal* 45.2 (2016). 235, pp. 235–269. ISSN: 0269-2821. DOI: 10.1007/s10462-015-9444-8. URL: https://doi.org/10.1007/s10462-015-9444-8.

[8] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[9] Wikipedia. *List of radio telescopes — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=List%20of%20radio%20telescopes&oldid=1109817148. [Online; accessed 22-November-2022]. 2022.

[10] W. N. Brouw. “Aperture Synthesis”. In: *Image Processing Techniques in Astronomy*. Ed. by C. De Jager and H. Nieuwenhuijzen. Dordrecht: Springer Netherlands, 1975, pp. 301–307. ISBN: 978-94-010-1881-4.

[11] van Haarlem, M. P. et al. “LOFAR: The LOw-Frequency ARray”. In: *A&A* 556 (2013), A2. DOI: 10.1051/0004-6361/201220873. URL: https://doi.org/10.1051/0004-6361/201220873.

[12] James J. Condon and Scott M. Ransom. *Essential Radio Astronomy*. en. Google-Books-ID: Jg6hCwAAQBAJ. Princeton University Press, Apr. 2016. ISBN: 978-1-4008-8116-1.

[13] S Yatawatta et al. “Initial deep LOFAR observations of epoch of reionization windows-I. The north celestial pole”. In: *Astronomy & Astrophysics* 550 (2013), A136.

[14] V Jelić et al. “Initial LOFAR observations of epoch of reionization windows-II. Diffuse polarized emission in the ELAIS-N1 field”. In: *Astronomy & astrophysics* 568 (2014), A101.

[15] George Heald, John McKean, Roberto Pizzo, et al. “Low Frequency Radio Astronomy and the LOFAR Observatory”. In: *Springer International Publishing, doi* 10 (2018), pp. 978–3.

[16] Rainer Beck, Marcus Brueggen, and Heino Falcke. “Lofar”. In: *Astronomische Nachrichten* 326.7 (2005), pp. 607–623. DOI: 10.1002/asna.200585007. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asna.200585007. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asna.200585007.

[17] AR Offringa et al. “A LOFAR RFI detection pipeline and its first results”. In: *arXiv preprint arXiv:1007.2089* (2010).

[18] Michael Mesarcik et al. “Deep learning assisted data inspection for radio astronomy”. In: *Monthly Notices of the Royal Astronomical Society* 496.2 (2020). 1517, pp. 1517–1529. ISSN: 0035-8711. DOI: 10.1093/mnras/staa1412. URL: https://doi.org/10.1093/mnras/staa1412.

[19] David R DeBoer et al. “Hydrogen epoch of reionization array (HERA)”. In: *Publications of the Astronomical Society of the Pacific* 129.974 (2017), p. 045001.

[20] Christian Jutten and Juha Karhunen. “Advances in blind source separation (BSS) and independent component analysis (ICA) for nonlinear mixtures”. In: *International journal of neural systems* 14.05 (2004), pp. 267–292.

[21] NRAO. *Very large array*. Mar. 2023. URL: https://public.nrao.edu/telescopes/VLA/.

[22] Wikipedia. *Westerbork Synthesis Radio Telescope — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Westerbork%20Synthesis%20Radio%20Telescope&oldid=1127679637. [Online; accessed 25-May-2023]. 2023.

[23] A Richard Thompson, James M Moran, and George W Swenson. *Interferometry and synthesis in radio astronomy*. Springer Nature, 2017.

[24] T.J. Dijkema. *LOFAR interferometry demonstration video*. [Video Presentation]. ASTRON, 2015.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[26] Martin Thoma. *A Survey of Semantic Segmentation*. 2016. DOI: 10.48550/ARXIV.1602.06541. URL: https://arxiv.org/abs/1602.06541.

[27] MA Syakur et al. “Integration k-means clustering method and elbow method for identification of the best customer profile cluster”. In: *IOP conference series: materials science and engineering*. Vol. 336. 1. IOP Publishing. 2018, p. 012017.

[28] M Rehbein, P Sahle, and T Schaßan. “Codicology and Palaeography in the Digital Age”. In: *Norderstedt: Schriften des Instituts für Dokumentologie und Editorik. Recuperado de http://kups.ub. uni-koeln. de/2939/el* 20.02 (2009), p. 2017.

[29] Giovanna Menardi. “Density-based Silhouette diagnostics for clustering methods”. In: *Statistics and Computing* 21.3 (2011), pp. 295–308.

[30] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[31] Andrew J Calder et al. “A principal component analysis of facial expressions”. In: *Vision Research* 41.9 (2001), pp. 1179–1208. ISSN: 0042-6989. DOI: https://doi.org/10.1016/S0042-6989(01)00002-5. URL: https://www.sciencedirect.com/science/article/pii/S0042698901000025.

[32] Pierre Comon. “Independent component analysis, a new concept?” In: *Signal processing* 36.3 (1994), pp. 287–314.

[33] Jan Melchior, Nan Wang, and Laurenz Wiskott. “Gaussian-binary restricted Boltzmann machines for modeling natural image statistics”. In: *PloS one* 12.2 (2017), e0171015.

[34] Abdulhamit Subasi and M Ismail Gursoy. “EEG signal classification using PCA, ICA, LDA and support vector machines”. In: *Expert systems with applications* 37.12 (2010), pp. 8659–8666.

[35] Alon Vinnikov and Shai Shalev-Shwartz. “K-means recovers ICA filters when independent components are sparse”. In: *International Conference on Machine Learning*. PMLR. 2014, pp. 712–720.

[36] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[37] Donald F Specht et al. “A general regression neural network”. In: *IEEE transactions on neural networks* 2.6 (1991), pp. 568–576.

[38] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. ISSN: 00189219. DOI: 10.1109/5.726791. URL: http://ieeexplore.ieee.org/document/726791/ (visited on 07/26/2023).

[39] D. Psaltis, A. Sideris, and A.A. Yamamura. “A multilayered neural network controller”. In: *IEEE Control Systems Magazine* 8.2 (Apr. 1988), pp. 17–21. ISSN: 0272-1708. DOI: 10.1109/37.1868. URL: http://ieeexplore.ieee.org/document/1868/ (visited on 07/26/2023).

[40] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[41] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[42] Sepp Hochreiter. “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 06.02 (Apr. 1998). Publisher: World Scientific Publishing Co., pp. 107–116. ISSN: 0218-4885. DOI: 10.1142/S0218488598000094. URL: https://www.worldscientific.com/doi/abs/10.1142/s0218488598000094 (visited on 07/27/2023).

[43] Sergio Bermejo and Joan Cabestany. “Oriented principal component analysis for large margin classifiers”. en. In: *Neural Networks* 14.10 (Dec. 2001), pp. 1447–1461. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(01)00106-X. URL: https://www.sciencedirect.com/science/article/pii/S089360800100106X (visited on 07/27/2023).

[44] Clark R. Givens and Rae Michael Shortt. "A class of Wasserstein metrics for probability distributions." en. In: *Michigan Mathematical Journal* (Jan. 1984). DOI: 10.1307/mmj/1029003026. URL: https://www.scinapse.io/papers/2040104067 (visited on 05/22/2023).

[45] Julien Rabin et al. "Wasserstein Barycenter and Its Application to Texture Mixing". In: *Scale Space and Variational Methods in Computer Vision*. Ed. by Alfred M. Bruckstein et al. Vol. 6667. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 435–446. ISBN: 978-3-642-24784-2 978-3-642-24785-9. DOI: 10.1007/978-3-642-24785-9_37. URL: http://link.springer.com/10.1007/978-3-642-24785-9_37 (visited on 05/22/2023).

[46] Sloan Nietert et al. *Statistical, Robustness, and Computational Guarantees for Sliced Wasserstein Distances*. en. arXiv:2210.09160 [cs, stat]. Oct. 2022. URL: http://arxiv.org/abs/2210.09160 (visited on 05/22/2023).

[47] Soheil Kolouri et al. *Generalized Sliced Wasserstein Distances*. en. arXiv:1902.00434 [cs, stat]. Feb. 2019. URL: http://arxiv.org/abs/1902.00434 (visited on 05/22/2023).

[48] I. Csiszar. "I-Divergence Geometry of Probability Distributions and Minimization Problems". In: *The Annals of Probability* 3.1 (Feb. 1975). ISSN: 0091-1798. DOI: 10.1214/aop/1176996454. URL: https://projecteuclid.org/journals/annals-of-probability/volume-3/issue-1/I-Divergence-Geometry-of-Probability-Distributions-and-Minimization-Problems/10.1214/aop/1176996454.full (visited on 05/23/2023).

[49] Quoc Le et al. "ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/233509073ed3432027d48b1a83f5fbd2-Paper.pdf.

[50] Kaiming He et al. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385 [cs]. Dec. 2015. DOI: 10.48550/arXiv.1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 07/29/2023).

[51] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. "Enhancing sparsity by reweighted 1 minimization". In: *Journal of Fourier analysis and applications* 14 (2008), pp. 877–905.

[52] Shibani Santurkar et al. "How Does Batch Normalization Help Optimization?" In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.

[53] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. "Design of an image edge detection filter using the Sobel operator". In: *IEEE Journal of solid-state circuits* 23.2 (1988), pp. 358–367.

[54] Mark A. Kramer. "Nonlinear principal component analysis using autoassociative neural networks". en. In: *AIChE Journal* 37.2 (1991). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690 pp. 233–243. ISSN: 1547-5905. DOI: 10.1002/aic.690370209. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209 (visited on 07/30/2023).

[55] Soheil Kolouri et al. *Sliced-Wasserstein Autoencoder: An Embarrassingly Simple Generative Model*. en. arXiv:1804.01947 [cs, stat]. June 2018. URL: http://arxiv.org/abs/1804.01947 (visited on 06/29/2023).

[56] J Macqueen. "SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS". en. In: *MULTIVARIATE OBSERVATIONS* (1967).

[57] S. Lloyd. “Least squares quantization in PCM”. en. In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489. URL: http://ieeexplore.ieee.org/document/1056489/ (visited on 07/30/2023).

[58] David Arthur and Sergei Vassilvitskii. “k-means++: The Advantages of Careful Seeding”. en. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), pp. 1027–1035.

[59] Charu C Aggarwal. “On k-anonymity and the curse of dimensionality”. In: *VLDB*. Vol. 5. 2005, pp. 901–909.

[60] Alexander Hinneburg and Daniel A. Keim. “Optimal Grid-Clustering : Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering”. In: *Proceedings of the 25 th International Conference on Very Large Databases, 1999*. 1999, pp. 506–517.

[61] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. “Finite Mixture Models”. en. In: (2019).

[62] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum Likelihood from Incomplete Data Via the EM Algorithm”. en. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1977.tb01600.x, pp. 1–22. ISSN: 2517-6161. DOI: 10.1111/j.2517-6161.1977.tb01600.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x (visited on 07/30/2023).

[63] Xu Yang et al. “Deep Spectral Clustering Using Dual Autoencoder Network”. en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 4061–4070. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00419. URL: https://ieeexplore.ieee.org/document/8953592/ (visited on 06/29/2023).

[64] Olatz Arbelaitz et al. “An extensive comparative study of cluster validity indices”. In: *Pattern recognition* 46.1 (2013), pp. 243–256.

[65] Peter J Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[66] Gideon Schwarz. “Estimating the Dimension of a Model”. In: *The Annals of Statistics* 6.2 (Mar. 1978). Publisher: Institute of Mathematical Statistics, pp. 461–464. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/aos/1176344136. URL: https://projecteuclid.org/journals/annals-of-statistics/volume-6/issue-2/Estimating-the-Dimension-of-a-Model/10.1214/aos/1176344136.full (visited on 06/29/2023).

[67] Harry Zhang. “The Optimality of Naive Bayes”. en. In: (2004).

[68] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. Aug. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[69] James Demmel, Ioana Dumitriu, and Olga Holtz. “Fast linear algebra is stable”. In: *Numerische Mathematik* 108.1 (2007), pp. 59–91.

[70] Zhidong Bai, Kwok Pui Choi, and Yasunori Fujikoshi. "Consistency of AIC and BIC in estimating the number of significant components in high-dimensional principal component analysis". In: *The Annals of Statistics* 46.3 (June 2018). ISSN: 0090-5364. DOI: 10.1214/17-AOS1577. URL: https://projecteuclid.org/journals/annals-of-statistics/volume-46/issue-3/Consistency-of-AIC-and-BIC-in-estimating-the-number-of/10.1214/17-AOS1577.full (visited on 06/28/2023).

[71] Yang You et al. *The Limit of the Batch Size*. en. arXiv:2006.08517 [cs, stat]. June 2020. URL: http://arxiv.org/abs/2006.08517 (visited on 06/27/2023).