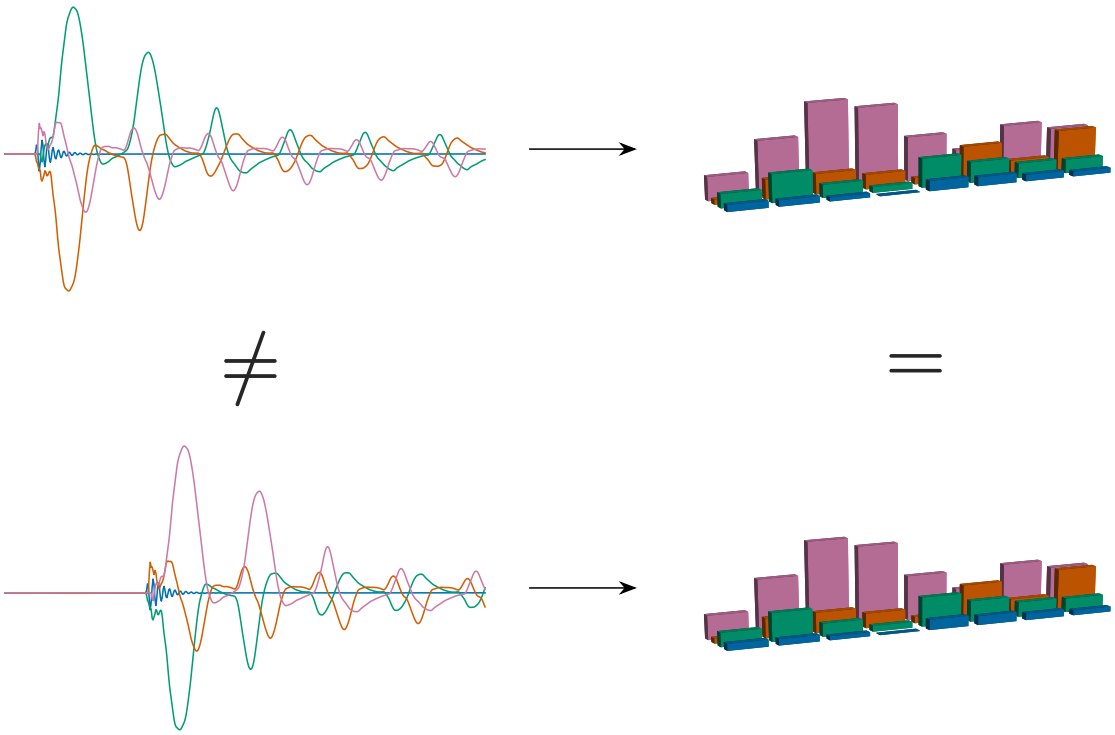

Invariant Feature Extraction for Power Quality Disturbances Using Deep Learning

Compact and Interpretable Features Improve Performance on Downstream Tasks

Master's thesis in
Systems, Control and Mechatronics,
Engineering Mathematics and Computational Science

Valter Schütz
Elias Stenhede Johansson

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Invariant Feature Extraction for Power Quality Disturbances Using Deep Learning

Compact and Interpretable Features Improve Performance on Downstream Tasks

Elias Stenhede Johansson
Valter Schütz

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Invariant Feature Extraction for Power Quality Disturbances Using Deep Learning
Compact and Interpretable Features Enhance Performance on Downstream Tasks
Valter Schütz, Elias Stenhede Johansson

Supervisor: Viktor Olsson, Eneryield
Examiner: Lars Hammarstrand, Electrical Engineering

Master's thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Time invariant and phase permutation invariant feature extraction.

Typeset in LaTeX
Gothenburg, Sweden 2024

Abstract

Feature extraction is a crucial step for tasks such as classification, clustering and predictive risk estimation. In power quality analysis, feature selection is often performed by domain experts. However, crafting a small yet sufficiently informative feature set is often challenging, particularly when evaluation data is scarce.

In this thesis, we present an autoencoder-based feature extraction method tailored for spectrograms of three-phase current and voltage waveforms. The extracted feature set is invariant with respect to time translation and sample length, and we demonstrate that sufficient information is retained to reconstruct the spectrogram with high accuracy. By applying Uniform Manifold Approximation and Projection (UMAP) with a specific pseudometric, invariance with respect to phase permutations can be obtained while further reducing the dimensionality of the extracted features.

The interpretability of the extracted feature sets is evaluated by observing changes in reconstructed signals when the latent variables are perturbed. The usefulness of the features is measured through three tasks: clustering, fault prediction and root cause disturbance classification. For the fault prediction task, we demonstrate that training an LSTM model with UMAP features significantly increases the AUC value compared to training the same model with manually selected features by a domain expert. In the classification task, both autoencoder features and UMAP features result in a higher macro F_1-score than manual features when training a neural network classifier, regardless of the training set size. The improvements are particularly notable for the smallest training sets. Additionally, using a model-free label propagation method on these features further enhances performance. Interestingly, we find that pretraining the autoencoder improves reconstruction fidelity, even when the pretraining dataset consists of audio recordings that are quite different from power quality measurements.

Keywords: Power Quality, Feature Extraction, Invariance, Deep Learning, Fault Prediction

Acknowledgements

First of all, we would like to thank Viktor Olsson and Jakob Lindqvist for all the valuable input and feedback they provided during the process of writing this thesis. Discussing ideas with them daily was both enjoyable and instructive. Our thanks also go to Filip Rydin for peer-reviewing the thesis and offering useful feedback for its improvement. We are grateful to Chalmers University of Technology for offering us a high-quality education and to the Swedish taxpayers for financing it.

Lastly, we express our gratitude to everyone at Eneryield for providing us with this thesis opportunity and a sociable work environment. Spinning the wheel for the Friday lunches was always one of the most eagerly anticipated events of the week.

Valter Schütz & Elias Stenhede Johansson, Gothenburg, May 2024

Acronyms & Nomenclature

$\mathcal{X}$	Power quality recording domain
$\mathcal{Z}$	Latent domain
$\mathcal{U}$	Embedding domain
X	Set of power quality recordings
Z	Set of latent vectors
U	Set of embedding vectors
x	Power quality recording
z	Latent vector
z_m	Manually engineered feature vector
u	Embedding vector

L	Loss function
J_e	Empirical risk
E_ϕ	Encoder
D_θ	Decoder

G	Graph
$\tilde{G}$	Fuzzy graph
E	Set of edges
V	Set of vertices
e	Edge
v	Vertex
$w_u, w_z, \hat{w}_u$	Membership functions

$L1$, $L2$, $L3$	Power system phases
N	Neutral phase
I	Current
U	Voltage
φ	Phase angle
σ	Power system phase permutation operator

DL	Deep Learning
FFT	Fast Fourier Transform
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
LSTM	Long Short-Term Memory
PCA	Principal Component Analysis
PQ	Power Quality
STFT	Short-Time Fourier Transform
UMAP	Uniform Manifold Approximation and Projection

Contents

1 Introduction

The electrical power grid, while designed for robustness and reliability, is not immune to failures. These failures can range from unpredictable events, such as accidental excavation of underground cables, to more predictable issues, such as wear and tear of power electronics. Moreover, the challenges are only increasing as the power grid evolves towards a more decentralised structure with more small-scale producers and more power electronics usage. While it is complicated to calculate the consequences of disturbances in the power grid, a frequently cited survey from 2007 estimates the cost in the EU alone to be €150 billion per year [39]. The study also finds that the general degree of monitoring is lacking, and that organisations taking fewer measurements are more likely to blame external factors such as neighbours and supply utilities, while good monitoring would have identified the problems as being their own.

The power grid operators that measure the voltage and current levels in their grid do so to ensure that the power delivered meets quality standards. Digital measurement devices require high sample rates to accurately capture transient events and harmonics. Since the amount of data collected is large, the need to reduce its size arises. Traditionally, this is done in one of two ways. Either by intermittently detecting and saving snippets of anomalous events or by periodically saving certain features. These include averages, minima, and maxima of specific quantities of interest that can be calculated from the data (e.g. harmonics, frequency, amplitude). If the intermittent method is used, the data will be high dimensional and noisy, taking up unnecessary amounts of space and making it difficult to train machine learning models with them. This motivates some form of dimensionality reduction method that extracts the most valuable information. Likewise, if the periodic method is chosen, one first needs to decide *which* features should be saved. As such, an alternative model that effectively extracts optimal features from raw waveforms would eclipse both of the former methods, especially if it is computationally simple enough to run online.

The aim of this project is to develop a feature extraction method that acts on current and voltage waveform data. Autoencoders trained using Deep Learning (DL) techniques have shown particular effectiveness in finding low-dimensional representations of data [15], and is therefore the architecture of choice in this thesis. Although the features produced by the autoencoder are compact and informative, it is difficult to directly learn a set of features that are phase-permutation invariant, which is why a second set of features is introduced, obtained by projecting the first set with Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [25].

After reducing the dimensionality of data, the informativeness of the two collections of extracted features is evaluated by using them for various downstream tasks such as root cause classification, fault forecasting, and visualisation. Using these tasks, the features are compared to features which are selected by a domain expert. Even though these tasks are used for evaluation in this thesis, it is important to note that features might have other future use cases. Thus, the features are not tailored to any specific task.

1.1 Background

In this section, we briefly describe the power grid, as well as the definition and relevance of power quality. Lastly, we give an overview of the use of DL on PQ data, as well as describe the context in which this thesis is written.

1.1.1 The power grid

The power grid connects electrical power producers with consumers, often transporting power across countries or even further. To efficiently transport large amounts of electrical power, high voltages are needed to reduce the losses produced by strong currents. Most transmission is carried out through overhead cables that are attached to electricity pylons as shown in Figure 1.1.

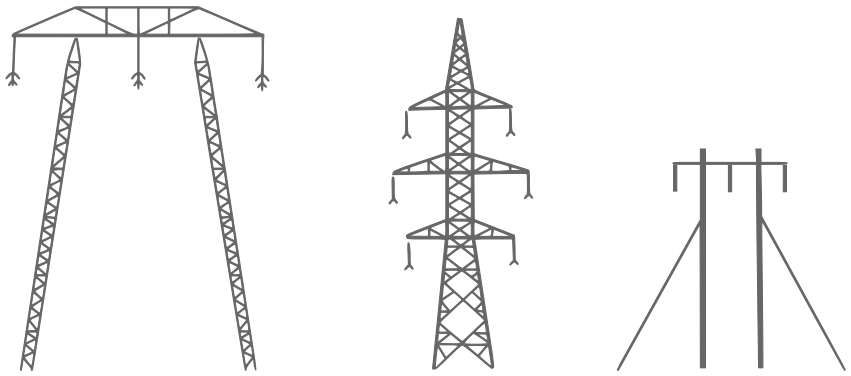

Figure 1.1: Three kinds of electricity pylons commonly seen in Scandinavia. For higher voltages, more robust constructions are needed.

However, high voltages are not suitable for consumers or producers, which is why transformer substations are installed throughout the network, increasing and decreasing voltage levels. For a simplified view of the power grid, see Figure 1.2. In high-voltage alternating current (AC) transmission lines, three conductors are used, each carrying its own sinusoidal current and voltage. They are called phases; ideally, they have the same voltage amplitude and are shifted 120° relative to each other. The frequency of the sinusoidal wave is 50 Hz in Europe, while a frequency of 60 Hz is used in North America and parts of South America. In distribution networks closer to end users, a fourth neutral conductor is used. The purpose of the neutral conductor is to provide a low impedance path to ground, reducing hazardous voltages that might otherwise be caused by imbalances and faults. Since the neutral current and voltage are typically zero, and the three phases, offset by 120 degrees, collectively add up to zero, the neutral conductor is often omitted. In such cases, the neutral channel can be effectively created by summing the three phases. Having a neutral conductor also makes the network suitable for a mixture of three-phase and single-phase (phase to neutral) loads, as is common in households. With a large number of customers on a grid, the single-phase loads are assumed to be somewhat evenly distributed across the phases so that the neutral current is small. In this thesis, we only work with 50 Hz grids, furthermore, the three phases will be referred to as L1, L2, and L3, while the neutral will be referred to as N. A typical waveform of the three phases and neutral during normal operation can be seen in Figure 1.3. During normal operation, the peak magnitudes of the currents might vary depending on demand, while the voltages are more stable.

1.1.2 Power quality

Much of the electrical equipment used requires constant voltage frequency and amplitude for efficient operation, and deviations might cause significant damage. It is therefore important to specify how the power grid is supposed to operate. Power Quality (PQ) refers to the degree to which voltages and currents behave as specified. It can be further divided into *quality of supply* and *quality of consumption*.

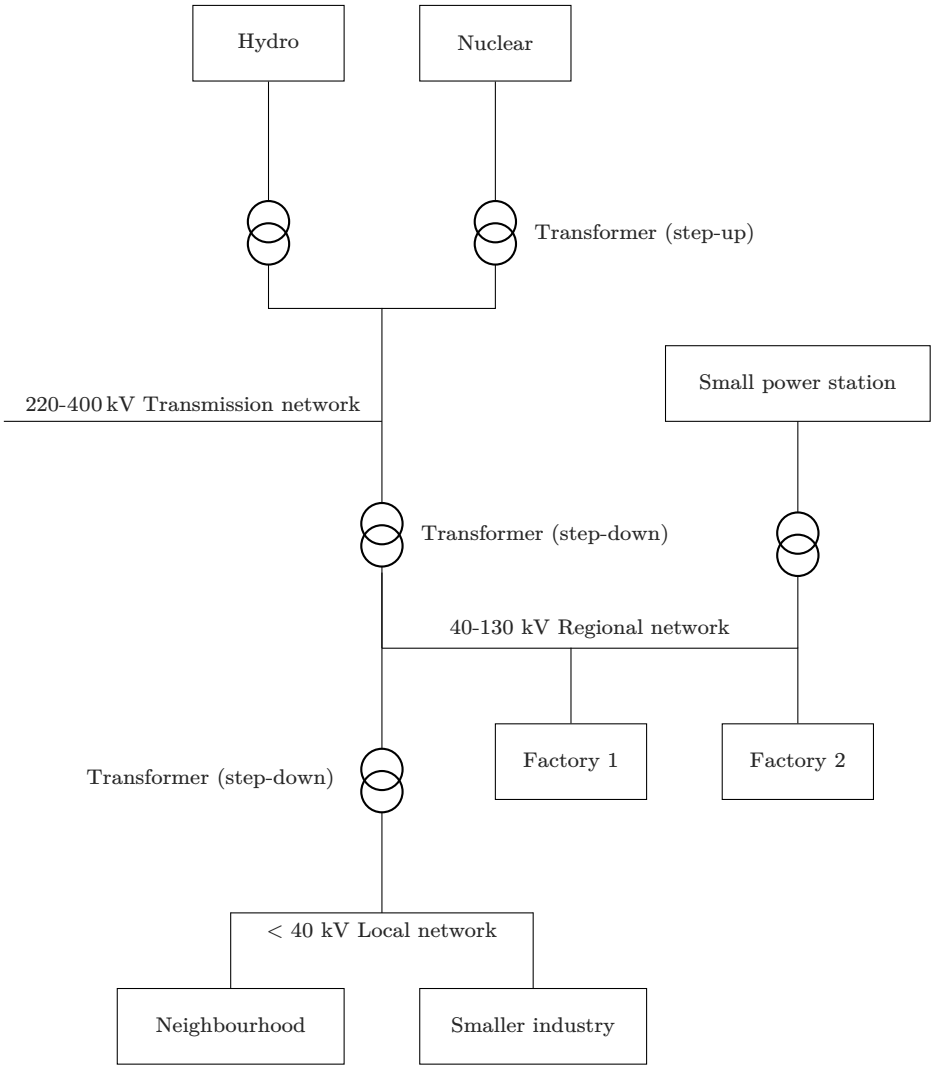

Figure 1.2: Sketch of a power grid with producers, consumers, transmission lines, and transformers.

The former is concerned with aspects that affect the power system as a whole, while the latter also takes into account phenomena that are only seen on the consumer's side [3]. A grid with good quality of supply would have

- Voltage stability.
- Frequency stability.
- Minimal distortions or harmonics of voltage waveforms.
- Few interruptions and other disturbances.

while quality of consumption would also require current and voltage to be in phase in order to reduce losses. While these points are quite unspecific, many requirements are rigorously defined in standards such as EN-50160 [5]. To name one requirement from EN-50160, the frequency should be between 49.5 Hz and 50.5 Hz 95 % of the time, averaged over 10 seconds.

PQ disturbances are generally sorted into two classes: variations and events. Vari-

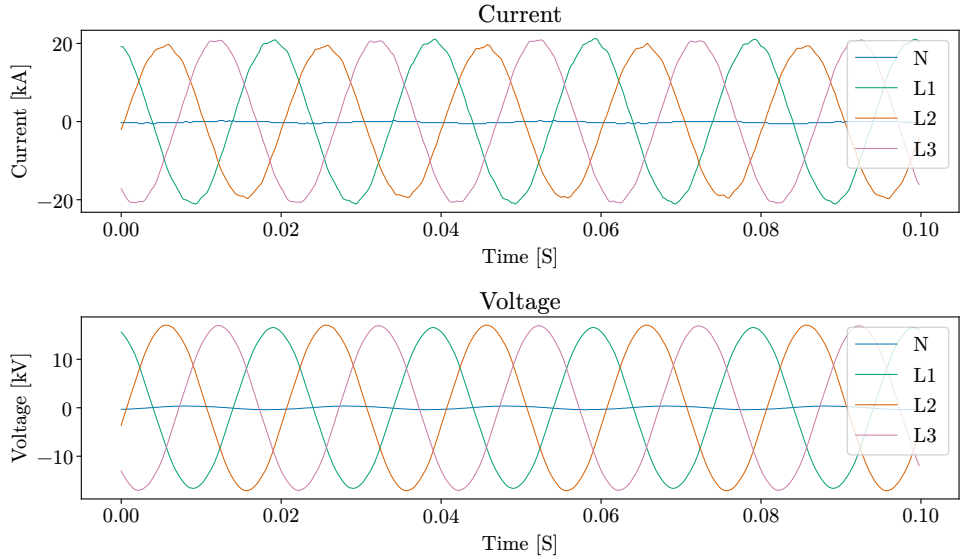

Figure 1.3: Current and voltage waveforms under normal conditions. The current waveform is not perfectly sinusoidal as it has some amount of harmonic distortion, which is expected.

ations are slight disturbances that are usually relatively unproblematic. Some variations include uneven load across the phases, harmonic voltage distortion, or slight voltage variations. Events are more serious, often sudden and transient disturbances that might damage equipment or disrupt operations. Perhaps the most common event studied in the literature is the voltage dip, which is shown in Figure 1.4 together with a transformer energisation event.

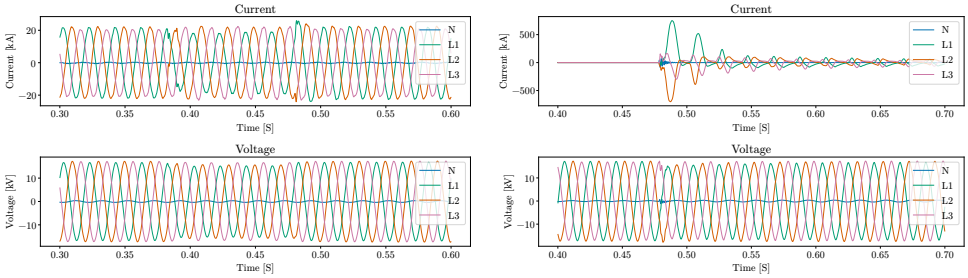

Figure 1.4: Waveforms during a voltage dip (left) and transformer energisation (right). The big currents to the right are called transformer inrush currents and occur when a transformer is connected to the grid.

Events such as short circuits, lightning strikes, and planned reconfigurations of the network might cause the voltage to deviate from specified levels. Another aspect affecting PQ is the loads that are connected to the system [5]. Changes in load magnitude, typically in a daily pattern, produce variations in the voltage amplitude. As noted before, from a quality of consumption perspective, the type of load connected is also relevant for reducing losses in the system. Inductive loads draw reactive power, which increases the losses in the system but can be counteracted by connecting capacitive loads. Examples of inductive components are electrical motors, transformers, and airborne transmission lines; examples of capacitive loads are buried cables and capacitor banks. Some loads might also introduce waveform distortions by drawing non-linear currents from the grid, for example, power electronics or heavy machinery such as arc furnaces. Power electronics approximate an analogue signal by quickly switching between discrete on- and off-states, and this naturally leads to upstream waveform distortions. As society transitions towards generating more energy by harvesting solar and wind power, as well as connects more electric

vehicle chargers, the waveform distortions due to power electronics will be steadily increasing.

1.1.3 Deep learning in power quality

With the abundant amounts of data collected by many PQ monitors, it is natural to ask whether DL can be applied to solve problems or gain insight. DL gained traction in the PQ research area around 2017 [29]. Common methods that were previously used include transformations such as wavelets and the s-transform, coupled with traditional ML methods such as support vector machines [11]. Most of the papers written about the applications of DL in PQ research concern the classification of disturbances. For example, [42] explores the use of convolutional networks for the classification of single-phase raw waveforms, using primarily synthetic data but also validating some single-phase PQ recordings. On the other hand, [21] utilises singular spectrum analysis together with the curvelet transform exclusively on synthetic data. In [28], an autoencoder is trained on voltage waveforms for a classification task with three classes, using real data. However, it is unclear if the model was trained using classification loss or reconstruction loss, making the versatility of the features questionable. As noted in [29], most research concerning PQ classification uses only synthetic single-phase data [43, 22, 23, 37, 32, 36, 6, 35, 7, 41, 19, 31, 38, 44], making it difficult to determine their efficacy on real data. Although some papers have trained a model on synthetic data and tested it on measured data, the performance is often poor [40, 4]. A major reason for the lack of models trained on real data is the lack of accessible open-source databases with labelled or unlabelled waveform data. Although [2] claims to be providing 1380 measurements, they cannot be accessed at the time of writing. In [31] and [16], labelled data is presented, but only for impulsive events and voltage dips, respectively.

Despite significant research efforts devoted to classification, its necessity remains unclear in some cases. As standards like IEEE 1159 [18], IEC 61000-4-30 [17] and EN 50160 [12] define the most common disturbance classes and many PQ monitors already have this knowledge built in, the question arises: is it even necessary to apply DL to this problem? If one only considers these rigorously defined classes, the answer is probably no. The necessity of a learnt model becomes clearer if one considers the task of root cause classification, i.e. the core of the issue rather than the recorded waveform, something that will be evaluated in this thesis.

There have also been some applications of DL on PQ data that are not related to classification. In [8], an autoencoder is used for feature clustering of spectral patterns and [14] applies a similar method for seeking daily patterns in harmonic distortion. In [33], an autoencoder is applied to three different types of input: waveforms, harmonic spectra, and superharmonic spectra. However, data is collected from railway pantographs rather than from the power grid, resulting in single-phase data and different characteristics. All of them use small and fully connected neural networks, but they still achieve satisfactory clustering of different types of disturbances in the latent space. This is surprising since the time invariance of disturbance classes suggests that other architectures are much more suitable for the task.

To the best of our knowledge, task independent autoencoders have not been applied to PQ disturbance waveforms collected from the electrical power grid. Furthermore, the concept of time invariance seems to be absent in current literature.

1.2 Feature extraction

The primary objective of this thesis is to perform automatic feature extraction on PQ waveform data. The main use cases for the extracted features are to train other statistical models, with the goal of performing tasks such as anomaly detection, root

cause classification, or fault prediction. Formally, the goal is that, for each high-resolution recording, the extracted features should be a vector $z \in \mathbb{R}^d$ with d as small as possible and z still containing as much information as possible, relevant for the downstream task. The motivation is partly to save memory but more importantly to facilitate the implementation of downstream statistical models. Evaluating the effectiveness of a feature set can be considered in terms of:

- **Informativeness** - The effectiveness of downstream models using these features is perhaps the most important factor. Although performance depends on the dataset, feature extraction process, and statistical model employed, the utility of a specific feature set can be assessed with a consistent dataset and model.
- **Interpretability** - If humans understand how events in real data influence the elements in z and the following output from a downstream statistical model, they are more likely to trust the system.
- **Compactness** - Having a low dimensional feature vector reduces the risk of downstream models overfitting.

When manual feature extraction is performed, one has to carefully determine which aspects of the data should be preserved; this also applies to a lesser extent when automatic feature extraction is performed. For example, any artefacts introduced by the measurement device should be ignored if possible.

1.3 Downstream tasks

Numerous potential downstream tasks could benefit from utilising the learnt features. This thesis will focus on evaluating the informativeness of the features by reconstruction performance, root cause classification, clustering, visualisation and to some extent fault risk estimation.

All deviating measurements are caused either by a problem with the measurement device or a root cause in the power grid. Examples of root causes are

- Short circuits, either at end-user or in the transmission system.
- Capacitor banks switching on or off.
- Network reconfigurations - new parts of the network connecting or disconnecting.
- Large machinery connecting or disconnecting.

Root cause classification is a harder task than classifying disturbances into classes defined by a standard since similar faults might not always result in similar recordings. For example, how a short circuit appears in the recording depends on multiple factors such as the distance from the fault, the speed at which protective devices are activated, and how other loads interact with the changing voltage.

Given a dataset collected in some power grid, it might be desirable to quickly visualise and inspect recent events. To make this as informative as possible, clustering and projecting the dataset onto a 2D surface might be of interest. Furthermore, clustering and visualisation might be useful for easier labelling of new data.

Forecasting the risk of serious disturbances is one of the most challenging yet crucial downstream tasks. This involves estimating the likelihood that a specific fault will occur within a given time frame, based on recent measurement data.

1.4 Description of datasets

In this thesis, two separate datasets are used, referred to as A and B. Both datasets contain waveform recordings of all three phases and neutral, but recorded at different parts of the grid and with different PQ monitors, see Table 1.1. Both datasets are proprietary and therefore anonymised; however, schematic sketches and some descriptions of the datasets are given.

Table 1.1: Comparison between dataset A and dataset B.

Dataset	**A**	**B**
Number of recordings	140000	2000
Grid frequency [Hz]	50	50
Nominal voltage [kV]	11	3.6 - 12.3
Sample rate [Hz]	4000	800 - 3200
Recording length [s]	1	0.25 - 3

1.4.1 Dataset A

Dataset A is collected by PQ monitors on two power substation bays. A bay is a parallel connection to the main cables connected directly to the downstream side of a transformer. PQ monitors are situated on two out of seven bays and save recordings whenever an abnormal event happens in any of the two bays. A schematic view of the substation is shown in Figure 1.5. Note that they share the same voltage, as they are situated in parallel.

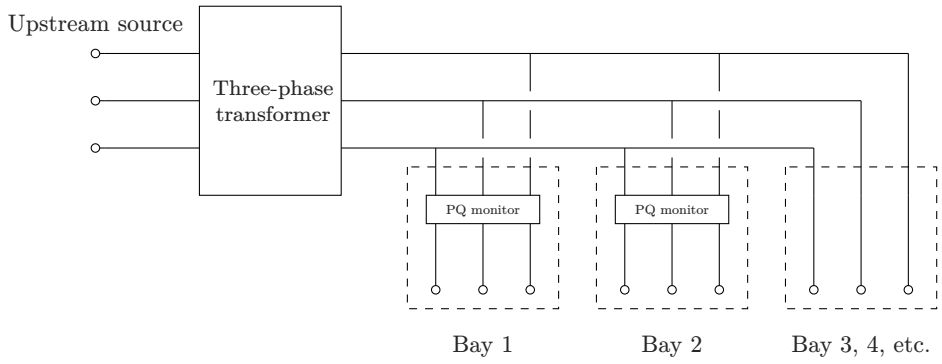

Figure 1.5: Measurement setup for dataset A. A three-phase transformer connected to a bus bar with two bays. Each bay is equipped with protective circuit breakers. At the breakers, both currents and voltages are recorded, however, they are only saved when a potential anomaly occurs.

As both PQ monitors save a waveform recording whenever an abnormal event is detected in one of them, they naturally form pairs, 140000 recordings correspond to 70000 unique timestamps. Furthermore, a subset of the points in dataset A has been labelled into eight classes representing root causes, identified in Table 1.2. Classes 0-3 represent unproblematic recordings while classes 4-7 represent more or less severe events. For some of the later results, the eight classes are collapsed into five to create plots which are easier to interpret. More specifically, classes 0 and 1 are collapsed into one class, and classes 4, 5 and 6 are collapsed into another.

Table 1.2: Labels used for evaluating feature informativeness.

Index	Description
0	No anomaly
1	Intermittent current swell
2	Network reconfiguration
3	Transformer energisation
4	Voltage dip
5	Intermittent fault
6	Serious fault
7	Material defect

1.4.2 Dataset B

Dataset B consists of recordings from several different locations, with 200-600 recordings per location. In contrast to dataset A, the measurements are taken at different transformers throughout a local grid, see Figure 1.6. The measurements are taken independently so faults local to a given transformer might only affect the local measurement while larger network errors might get recorded from several locations. Clearly, dataset A is much larger which makes it more suitable for training an autoencoder. Dataset B is however used to examine how well the parameterised feature extraction method works across domains, by employing transfer learning in Section 4.5.

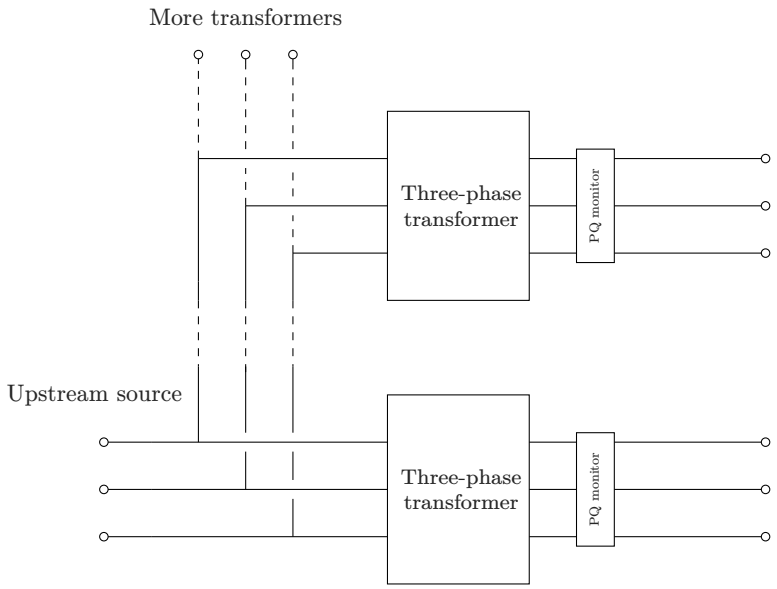

Figure 1.6: Measurement setup for dataset B. There are several different three-phase transformers, each with a PQ monitor installed on the low-voltage side. Just as in the case of dataset A, measurements are only taken when a potential anomaly occurs.

1.5 Manually engineered features

The primary benchmark features considered in this thesis have been hand-crafted by a domain expert through the process of computing informative scalar values from PQ related quantities such as impedance, power, current and voltage. Examples

include minimum and maximum values, as well as standard deviations and integrated values of for example harmonics and transients. Besides being selected for being informative, feature selection has also been applied, reducing the dimensionality by excluding highly correlated features. In total, a few hundred scalar values are extracted for each waveform. The manually engineered feature set is used for forecasting in [1].

2 Theory

This section begins with presenting the mathematical details behind training an autoencoder, followed by a discussion of equivariance and invariance, and a theoretical justification for UMAP. Lastly, an algorithm for belief propagation on fuzzy graphs is presented.

2.1 Feature extraction using autoencoders

Let $\mathcal{X} \subset \mathbb{R}^{C \times T}$ denote the set of all possible recordings at some location. The dataset $X = \{x_1, \dots, x_N\} \subset \mathcal{X}$ consists of N recordings with C channels and T observations in time. An autoencoder is a pair of parameterised functions: an encoder $E_\phi : \mathcal{X} \mapsto \mathcal{Z}$ and a decoder $D_\theta : \mathcal{Z} \mapsto \mathcal{X}$, where $\mathcal{Z} \subset \mathbb{R}^m$ and $m < C \cdot T$ is called the latent dimension. The joint task of E_ϕ and D_θ is to compress and then reconstruct samples from the dataset. The quality of the reconstruction is measured by a loss function

$$L\left\{x_i, D_\theta(E_\phi(x_i))\right\}.$$

Squared Euclidean distance (L_2) and Manhattan distance (L_1) are the most common loss functions for training autoencoders. In this thesis, Manhattan distance will be used.

The optimal parameters ϕ^*, θ^* for the autoencoder are the parameters that minimise the *risk*

$$\mathbb{E}\left[L\{x, D_\theta(E_\phi(x))\}\right]. \tag{2.1}$$

However, $\mathcal{X}$ is unknown, and we only have access to measured samples of the data distribution, so we approximate Equation (2.1) with the *empirical risk*

$$J_e(\phi, \theta; X) \doteq \frac{1}{N} \sum_{x_i \in X} L\{x_i, D_\theta(E_\phi(x_i))\}. \tag{2.2}$$

The empirical risk J_e can be minimised using gradient descent, or more commonly batched Stochastic Gradient Descent (SGD). Letting α denote a sufficiently small learning rate, and X_B denote a random subset of X with B elements, the parameters are updated as

$$\begin{bmatrix}\phi \\ \theta\end{bmatrix} \longleftarrow \begin{bmatrix}\phi \\ \theta\end{bmatrix} - \alpha \begin{bmatrix}\nabla_\phi J_e(\phi, \theta; X_B) \\ \nabla_\theta J_e(\phi, \theta; X_B)\end{bmatrix}.$$

As X is often large, using SGD is much more efficient than calculating the gradient for the whole dataset. Furthermore, using SGD yields better generalisation performance in general. In high dimensional settings, algorithms such as Adam [20] often converge even faster and are less sensitive to the choice of α. Adam builds on the same foundations as SGD and is not further explained here.

If the autoencoder is well trained, that is, has a sufficiently small loss value for every recording in X, then every recording $x \in X$ will correspond to a unique feature vector $z = E_\phi(x)$. The characteristic vector z is also commonly called the *latent vector*, and will be informative and compact if $m \ll C \cdot T$. It might not be very interpretable, however, which is why restrictions are imposed on the autoencoder architecture, as described in Section 3.2.

2.2 Invariance and equivariance

For three-phase disturbance recordings, many characteristics of the recordings depend only on the measurement device, rather than the power grid being measured. Some examples include

- Sample rate.
- Recording length.
- Quantisation, the minimum resolution of currents and voltages.
- Phase naming, the ordering of the phases.
- Disturbance timing, recorded with varying margins before the event.

Since we are only interested in the underlying system, we would ideally have features which do not depend on the settings of the measurement device. That is, we want to construct models which are invariant with respect to such external characteristics.

Definition 1. *Let T and F be two functions, we say that F is **invariant** under T if*

$$F(T(x)) = F(x) \tag{2.3}$$

*and **equivariant** if*

$$F(T(x)) = T(F(x)). \tag{2.4}$$

The easiest way to illustrate the applicability of invariance in feature extraction is through an example.

Example 1. *The encoder E_ϕ would be invariant with respect to both time translation and phase permutation if*

$$E_\phi(x) = E_\phi(T_t(\sigma(x))) = z \tag{2.5}$$

where T_t and σ represent the operators for time translation and phase permutation respectively, see Figure 2.1 for an illustration.

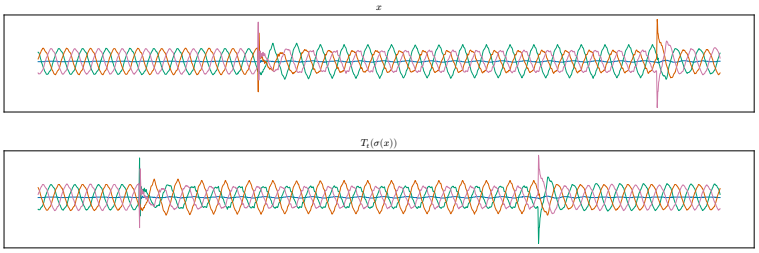

Figure 2.1: At the top of the figure, a recording $x \in X$. At the bottom, the same recording is shown after being translated in time and permuted. If both signals would result in the same latent vector, the encoder would be invariant to time translation and phase permutation.

2.3 Uniform manifold approximation and projection

Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that embeds high-dimensional data in a lower dimension by identifying a topological graph in the high-dimensional space and tries to find a similar embedding

of the graph in the lower dimension. Introducing UMAP serves two purposes: reducing dimensionality and obtaining invariance with respect to phase permutations. Further reducing dimensionality from the latent vector z might help certain downstream tasks, and choosing the embedding dimension 2 or 3 is ideal for visualisation. UMAP relies on three basic assumptions, namely

(1) The data is uniformly distributed on some (likely unknown) Riemannian manifold (M, g).

(2) The Riemannian metric g can be approximated as locally constant.

(3) The manifold M is locally connected.

The assumption (3) that real signal data exists on a locally connected manifold is generally accepted because minor modifications to the signal still produce a realistic result. However, assumptions (1) and (2) are not immediately clear. Therefore, the subsequent subsections will present the necessary theory to explain how to validate them. The definitions are found in [13, 26] and Lemma 1 from [25]. If one wishes to learn more about simplicial complexes and algebraic topology in general, [13, 26] can be used as sources. Readers less inclined towards abstract mathematics can proceed directly to Section 2.3.4, where the practical aspects of UMAP are discussed.

2.3.1 Riemannian manifolds

In this section, we introduce the concept of Riemannian manifolds. They are vital for the UMAP algorithm as they provide a mathematical framework for understanding the geometry of high-dimensional data.

Definition 2. *Let M be a smooth real manifold, and let T_pM denote the tangent space at $p \in M$. Let $g_p : T_pM \times T_pM \to \mathbb{R}$ denote a positive-definite bilinear function depending smoothly on p. Such a function is called a Riemannian metric, and (M, g) is called a* ***Riemannian manifold****.*

To give a bit more insight into what Riemannian manifolds might look like, a few examples are provided.

Example 2. *Standard n-dimensional Euclidean space $\mathbb{R}^n$ is a Riemannian manifold. Given any point $p \in \mathbb{R}^n$, the tangent space is the vector space containing all directions in which it is possible to pass tangentially through p. In this case, the tangent space is $\mathbb{R}^n$ itself. Although the tangent space T_pM always has the same dimension as M, it is not always M itself as in this example. As for the Riemannian metric on this manifold, the Euclidean metric can be used, which is independent of p.*

Example 3. *The surface of a sphere in $\mathbb{R}^3$ is a two-dimensional manifold, having as tangent spaces the plane passing through p tangentially to the sphere, see Figure 2.2. The Riemannian metric in this case can be inherited from the Euclidean metric of $\mathbb{R}^3$, restricted to tangent vectors at p.*

Although these examples were easy to visualise, connected (in one piece), and had a directly inherited Euclidean metric, Riemannian manifolds do not need to be either completely connected or directly inherit their metric from the ambient Euclidean space. To highlight this, a final example is presented.

Example 4. *Let the (not necessarily parameterisable) manifold M be some smooth m-dimensional structure embedded in $\mathbb{R}^n$, where $m < n$. Furthermore, let g'_p and g''_p be two Riemannian metrics on the manifold and $\lambda : M \mapsto [0, 1]$ some $\mathcal{C}^1$ function. Letting $g_p = \lambda(p)g'_p + (1 - \lambda(p))g''_p$, it follows that g_p is a Riemannian metric since $\lambda \in [0, 1]$. In this case, g_p is not directly inherited from the surrounding Euclidean space.*

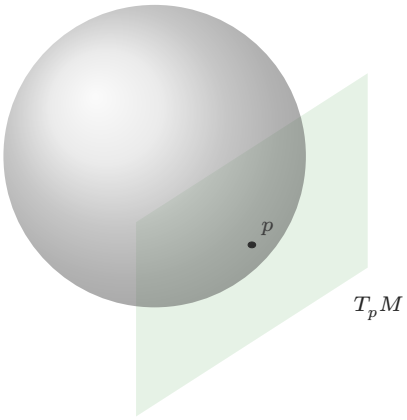

Figure 2.2: The surface of a sphere embedded in three dimensions can be thought of as a two dimensional Riemannian manifold. At each point p on the surface, a metric is defined on the tangent space.

Having a metric g which depends on the point p adds flexibility; for example, we can design a metric g so that the data distribution is uniform with respect to g. For this to be useful however, we need to be able to find the relation between g_p and the metric inherited from $\mathbb{R}^n$. It turns out that Lemma 1 provides exactly this.

Lemma 1. *Let (M, g) be a Riemannian manifold in the ambient $\mathbb{R}^n$, and let $p \in M$ be a point. If g is locally constant about p in an open neighbourhood $U \subseteq M$ such that g is a constant diagonal matrix in the ambient coordinates, then in a ball $B \subseteq U$ centred at p with volume $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ with respect to g, the geodesic distance from p to any point $q \in B$ is $d_{\mathbb{R}^n}(p, q)/r$, where r is the radius of the ball in the ambient space and $d_{\mathbb{R}^n}$ is the existing metric in ambient space.*

To get an intuitive understanding of what Lemma 1 says, note that B is the n-dimensional unit ball with respect to g centred at p. Furthermore, the measure g is a constant diagonal matrix on B, which means that it must be proportional to the ambient metric $d_{\mathbb{R}^n}$. By letting r denote the radius of B with respect to $d_{\mathbb{R}^n}$ and noting that the radius with respect to g is 1, it is clear that the relationship between the two is precisely r. For a proof of Lemma 1, see Appendix A in [25].

We have now introduced the concept of Riemannian manifolds, however, the manifold which we assume the data is sampled from is unknown. We thus need some way to estimate the shape of the manifold using a dataset Z, which in our results will represent autoencoder features. To do this, we need to introduce the concept of simplices and simplicial complexes.

2.3.2 Simplices and simplicial complexes

As simplicial complexes are sets of simplices, we will start by defining these smaller building blocks.

Definition 3. *Let $\{a_0, \dots, a_n\}$ be a geometrically independent set in $\mathbb{R}^n$. Define the **n-simplex** σ spanned by $a_0, \dots, a_n$ to be all points $z \in \mathbb{R}^{n_z}$ such that*

$$z = \sum_{i=0}^{n} c_i a_i \quad \textit{where} \quad \sum_{i=0}^{n} c_i = 1 \quad \textit{and} \quad c_i \geq 0 \quad \forall i. \tag{2.6}$$

Note that the number n in n-simplex represents the dimensionality of the geometrical object spanned by the $n+1$ points $\{a_0, \dots, a_n\}$. Examples of 0, 1 and 2-simplices are shown in Figure 2.3.

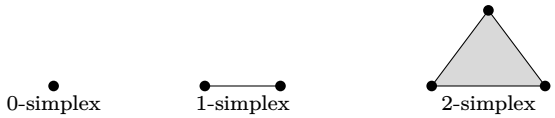

Figure 2.3: 0-simplices, 1-simplices and 2-simplices are points, line segments and triangles respectively.

We call any simplex spanned by a subset of $\{a_0, \dots, a_n\}$ a *face* of σ. Using the concept of a face, we can define a simplicial complex.

Definition 4. *A **simplicial complex** K in $\mathbb{R}^n$ is a set of simplices in $\mathbb{R}^n$ such that*

1. *Every face of a simplex in K is also in K.*
2. *The intersection of any two simplices in K is a face in each of them.*

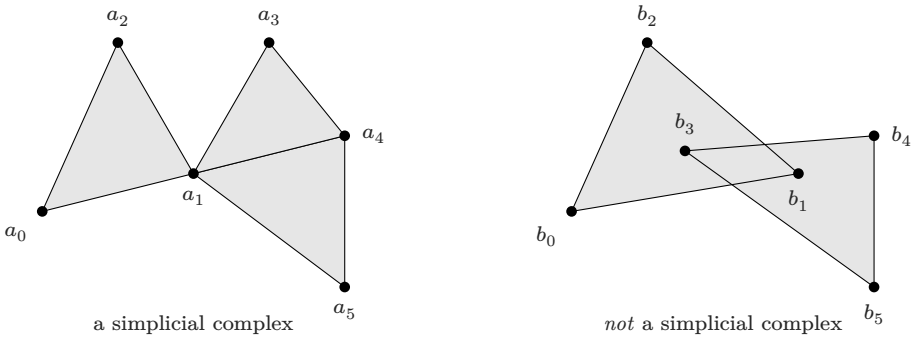

Figure 2.4: Left: a simplicial complex with 6 vertices. Right: a set of triangles that is not a simplicial complex, since there exists an intersection that is not a face in each of the triangles, violating condition 2.

In other words, a simplicial complex is a set of points, line segments, triangles and their higher-dimensional counterparts, with certain restrictions on what has to be in the set; see Figure 2.4. For our purposes, we need to pick out subsets of simplicial complexes, which is why we introduce the p-skeleton.

Definition 5. *The **p-skeleton** $K^{(p)}$ of the simplicial complex K is a subset of K that contains all simplices with dimension at most dimension p.*

The elements in $K^{(0)}$ are called the *vertices* of K. The next step is to construct a simplicial complex given an arbitrary dataset, with the goal of capturing the local topology. To achieve this, we introduce the concept of a *Čech Complex*. This construction is particularly useful for understanding the shape or topology of datasets.

Definition 6. *Let $Z = \{z_i\}_{i=1}^N$ be a dataset and d some distance function so that (Z, d) is a metric space. Let σ_I denote the simplex spanned by $I \subseteq Z$. Furthermore, let $r \in \mathbb{R}_+$ be a scale parameter and $B_r(z)$ the ball with radius r centred z. The **Čech complex** of Z with radius r is then defined as*

$$\mathcal{C}_r(Z) \doteq \left\{ \sigma_I \middle| I \subseteq Z \text{ and } \bigcap_{z_i \in I} B_r(z_i) \neq \emptyset \right\}. \tag{2.7}$$

For a graphical explanation, see Figure 2.5, and note that simplices are only created where balls intersect. Although there is no limit on the dimension of the simplices in a Čech complex, the UMAP algorithm only needs the 1-skeleton $\mathcal{C}_r^{(1)}(Z)$ of the complex to yield good embedding results, while also being much more computationally efficient.

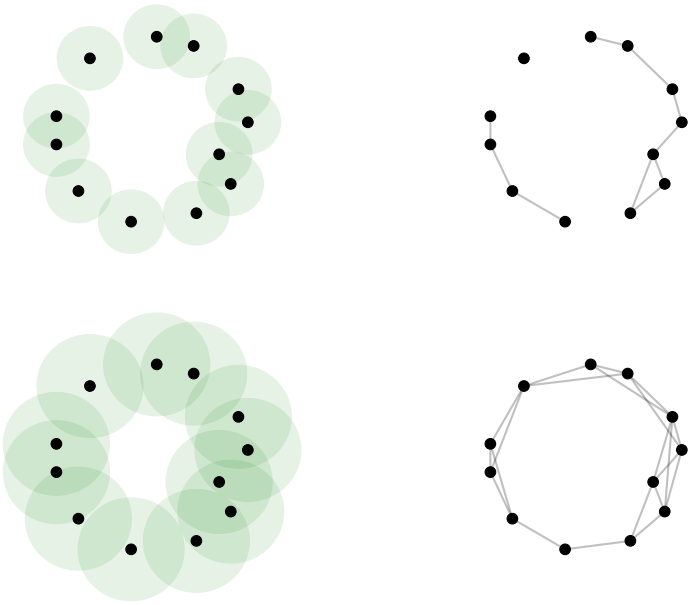

Figure 2.5: Illustration of how Čech complexes are generated for two different scale parameters r. The complexes contain simplices wherever the distance between two points is smaller than $2r$. In this figure, only the 0- and 1-simplices are displayed.

Looking at Figure 2.5, observe that the graph can be partially disconnected or have a very high number of connections, depending on the scale parameter. Having a disconnected point implies a lack of topological information, making it impossible to find a good embedding using this representation. Furthermore, the task of choosing a good scale parameter is impossible in higher dimensions, because of the curse of dimensionality (for some fixed radius, there would be a very high probability of either having a very disconnected graph, or an almost fully connected graph). The problem can however be resolved by making use of the flexible definition of Riemannian metrics. As indicated in Example 4, a separate distance function can be defined for each $z_i \in Z$, with the goal of ensuring that each ball centred at z_i should contain approximately k neighbouring points, satisfying the uniform density assumption of UMAP. See Figure 2.6 for an illustration; further details of the implementation will be discussed in Section 2.3.4.

Although we are now closer to finding a good representation for the dataset, it is still possible for two points z_i and z_j to be within a certain radius with respect to the metric defined on $T_{z_i}M$ but not the metric defined on $T_{z_j}M$. In other words, we have managed to fulfil assumptions (1) and (2) for UMAP, but now have a set of N metric spaces which are incompatible. To overcome this discrepancy, fuzzy sets are introduced.

2.3.3 Fuzzy sets

In fuzzy sets, as opposed to traditional (crisp) sets, the degree of membership of an element in the set can range between 0 and 1, with any value in between representing

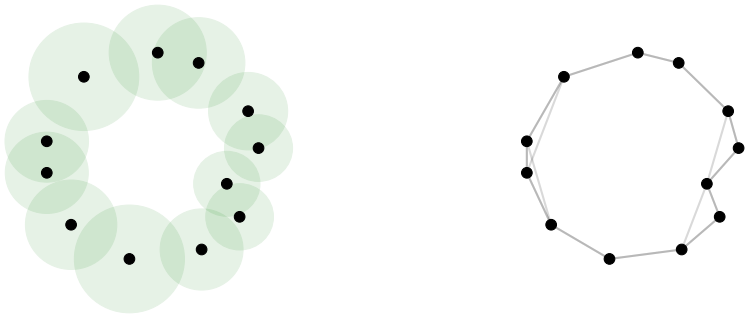

Figure 2.6: Illustration of how a Čech complex is generated using a non-constant metric. Now, the radii are not constant with respect to the ambient metric.

degrees of membership. This allows assigning degrees of membership (which might be interpreted as probabilities of existence) to each simplex in a Čech complex.

Definition 7. *A fuzzy set (A, μ) is a set of ordered pairs, each pair consisting of an element a from a set A and the degree of membership $\mu(a)$ of a in (A, μ)*

$$(A, \mu) \doteq \{(a, \mu(a)) \mid a \in A\}. \tag{2.8}$$

For the 1-skeleton Čech complex used by UMAP, our primary concern is what degree of membership should be assigned to the edges (1-simplices). In this case, A would be the set of all edges, and the membership function μ should be designed so that each point z_i in the dataset has at least one edge that is fully included ($\mu = 1$) in order to prevent isolated points, as illustrated in Figure 2.7. The membership function μ should also decrease as the distance between points increases. To construct such a fuzzy set, a more relaxed notion of distance is needed.

Definition 8. *A **pseudometric space** (Z, d) is a set Z together with $d : Z \times Z \to \mathbb{R}_+$, called a pseudometric, such that for every $z_i, z_j, z_k \in X$,*

1. *Distance to self: $d(z_i, z_i) = 0$*
2. *Symmetry: $d(z_i, z_j) = d(z_j, z_i)$*
3. *Triangle inequality: $d(z_i, z_k) \leq d(z_i, z_j) + d(z_j, z_k)$*

The point of this is to be able to define the distance between nearest neighbours as 0, even though they are not the same point. This is not allowed for metric spaces, as it is required that the distance between separate points be strictly positive.

The last step in the representation is to make μ symmetric. Let $a_{ij} \in A$ and $a_{ji} \in A$ be the edges from z_i to z_j and vice versa. A new fuzzy set (A, ν) can now be defined that has the same set A, but a new membership function ν, defined as

$$\nu(a_{ij}) = \nu(a_{ji}) = \mu(a_{ij}) + \mu(a_{ji}) - \mu(a_{ij})\mu(a_{ji}). \tag{2.9}$$

If μ is interpreted as measuring the probability of an edge existing, ν now measures the probability of either a_{ij} or a_{ji} existing.

The dataset representation is now complete. To embed (A, ν) in some lower dimensional space (for example $\mathbb{R}^2$), a new membership function ν' can be defined on a preliminary projection of the data. The cross entropy between two fuzzy sets then needs to be minimised which will affect the projection.

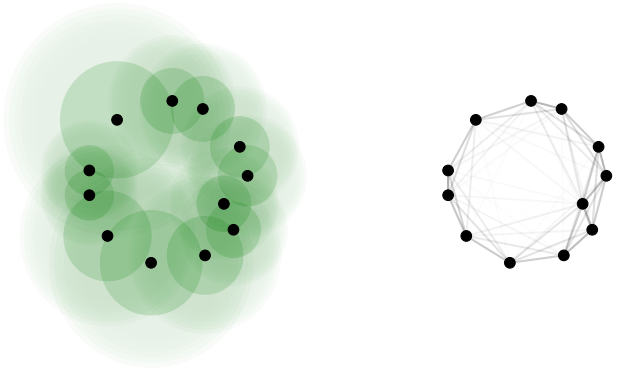

Figure 2.7: Each point in the dataset is endowed with a pseudo-metric, resulting in a fuzzy Čech complex shown to the right, where membership strength is indicated by the opacity of the edges. Note that each vertex has one edge that is fully opaque ($\mu = 1$). Only the 1-skeleton of the fuzzy Čech complex is shown for simplicity.

Definition 9. *The cross entropy C of two fuzzy sets (A, ν) and (A, ν') is defined as*

$$C((A,\nu),(A,\nu')) \doteq \sum_{a \in A} \left(\nu(a) \ln \left(\frac{\nu(a)}{\nu'(a)} \right) + (1 - \nu(a)) \ln \left(\frac{1-\nu(a)}{1-\nu'(a)} \right) \right). \quad (2.10)$$

Minimisation can be done using gradient-based optimisation; details will be discussed in the following section.

2.3.4 Algorithmic description

With the basic foundations of UMAP covered, we turn to the actual implementation. The embedding of some dataset Z into a lower dimensional space consists of two main steps:

1. Approximating the topology of the dataset by creating a fuzzy graph where neighbouring points have edges with high degrees of membership. By fuzzy graph, we mean that the fuzzy set contains edges.

2. Finding a corresponding fuzzy set in a lower dimension that minimises the cross-entropy with the fuzzy set found in the first step.

We describe the procedure in the same order as it is performed.

2.3.4.1 Creating a fuzzy graph representation

Let $(\mathcal{Z}, d)$ denote a pseudometric space where $Z \subset \mathcal{Z}$ is a representation of a dataset, e.g the latent vectors obtained by passing the original dataset X through the function E_ϕ as described in Section 2.1. The first step consists of fixating the hyperparameter k and constructing a k-nearest neigbour graph (k-NNG) denoted $G = (Z, E)$, where $E \subseteq \{(z_i, z_j) | z_i, z_j \in Z \text{ and } i \neq j\}$, and $(z_i, z_j) \in E$ is denoted as e_{ij}. The k-NNG can be constructed exactly for small datasets, or approximately using more efficient methods, see e.g. [10].

We now switch perspectives towards interpreting G as the 1-skeleton of a Čech complex. Recall from Definition 6 that the edges are included in the complex if the distance between two points is less than $2r$ with respect to some metric. We now

want to infer the metric given the edges. For every point z_i, let $\mathcal{N}_i$ denote the set containing z_i's k nearest neighbours, and define the parameter ρ_i as the distance to its closest neighbour (not including itself)

$$\rho_i \doteq \{\min(d(z_i, z_j))|z_j \in \mathcal{N}_i, d(z_i, z_j) > 0\} \tag{2.11}$$

and the local scale parameter σ_i as the solution to

$$\sum_{z_j \in \mathcal{N}_i} \exp\left(\frac{-\max(0, d(z_i, z_j) - \rho_i)}{\sigma_i}\right) = \log_2(k). \tag{2.12}$$

The parameters ρ_i and σ_i together define the Riemannian metric around z_i.

We can now form a fuzzy set of directed edges $\tilde{G} = (Z, E, w)$ by taking the k-NNG and endowing it with a weight function of the edges defined as

$$w_z(z_i, z_j) = w_z(e_{ij}) \doteq \begin{cases} \exp\left(\frac{-\max(0, d(z_i, z_j) - \rho_i)}{\sigma_i}\right), & \text{if } z_j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

Note that for every $z_i \in Z$ the membership function $w(e_{ij}) = 1$ for one or more values of j. The fuzzy set of edges which is now obtained is not symmetric, i.e. $w(e_{ij}) \neq w(e_{ji})$ in general. To obtain a symmetric fuzzy set, we apply Equation (2.9) which concludes the construction of the fuzzy graph that represents our dataset, hereafter denoted by $\tilde{G}_z = (Z, E, w_z)$ where w_z denotes the symmetric membership function.

2.3.4.2 Embedding the fuzzy graph

For the second step, we need to choose a domain in which to embed $\tilde{G}_z$. We denote the embedding domain $\mathcal{U}$; a suitable domain is, for example, $\mathbb{R}^2$ since it allows for visualisation. Let $U \subset \mathcal{U}$ be a preliminary embedding of Z, and let $\tilde{G}_u = (U, E, w_u)$ be the embedded fuzzy graph. Intuitively, it makes sense to have w_u be of the same form as w_z defined by w in Equation (2.13), but with a constant scale parameter. In other words, we want

$$w_u(u_i, u_j) = \begin{cases} 1 & \text{if } ||u_i - u_j|| \leq \rho \\ \exp(\rho - ||u_i - u_j||) & \text{otherwise} \end{cases} \tag{2.14}$$

where ρ is a hyperparameter often called *min-dist*. However, since we want to minimise cross-entropy using gradient descent-based methods, we instead choose the membership function as a smooth approximation of w_u, more specifically, we choose

$$\hat{w}_u(u_i, u_j) = \hat{w}_u(e_{ij}) \doteq (1 + \alpha||u_i - u_j||_2^{2\beta})^{-1} \tag{2.15}$$

and fit the parameters α and β, see Figure 2.8.

After initialising the embedding U, the next step is to minimise the cross-entropy between (E, w_z) and $(E, \hat{w}_u)$ as defined in Equation (2.10)

$$L = \sum_{e \in E} \left(w_z(e) \ln\left(\frac{w_z(e)}{\hat{w}_u(e)}\right) + (1 - w_z(e)) \ln\left(\frac{1 - w_z(e)}{1 - \hat{w}_u(e)}\right)\right). \tag{2.16}$$

The membership function w_z is fixed during the optimisation, so the equation simplifies to

$$L = -\sum_{e \in E} \left(w_z(e) \ln \hat{w}_u(e) + (1 - w_z(e)) \ln(1 - \hat{w}_u(e))\right). \tag{2.17}$$

The first term ensures that if $d(z_i, z_j)$ is small, $||u_i - u_j||_2$ is small, and the second term ensures that if $d(z_i, z_j)$ is large, $||u_i - u_j||_2$ is large. In other words, the distance

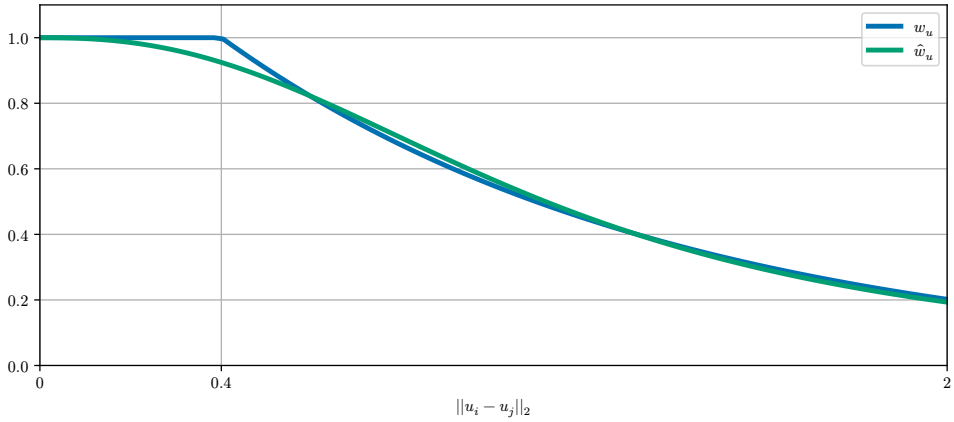

Figure 2.8: The ideal membership function w_u and smooth membership function $\hat{w}_u$, defined on the embedding. In this example, the parameter *min-dist* is set to 0.4.

is preserved between the high-dimensional data and the embedding. The last step consists of minimising Equation (2.17), for example by using a batched sampling approach. For each step, we construct two batches of edges, E_B^1 and E_B^2. Edges in E_B^1 are sampled with probability proportional to their membership strength in the original graph $w_z(e)$, and edges in E_B^2 are sampled uniformly from E. The loss to minimise is now

$$\hat{L} = -\sum_{e \in E_B^1} \ln \hat{w}_u(e) + \sum_{e \in E_B^2} \ln(1 - \hat{w}_u(e)).$$

For computational reasons, we have approximated $w_z(e) \approx 0$ in the second term, as the number of data points is significantly larger than the number of nearest neighbours, meaning most sampled edges will actually have $w_z(e) = 0$. Intuitively, one can think of the first term as forcing neighbours to stay close, while the negatively sampled second term makes the embedding points $u \in U$ repel each other.

2.4 Feature evaluation using classification

One way to evaluate the informativeness of extracted features is classification. For this we need both a classification method and some method to evaluate the results. We use a neural network and a model-free method; the model-free method is presented below.

2.4.1 Belief propagation on fuzzy graphs

Belief propagation operates on graphs and is based on Bayes' theorem and the assumption that neighbouring points are likely to belong to the same class. It is a message-passing algorithm, where graph nodes are only influenced by their neighbours in each time step. Since the method is only used for feature evaluation, we only try to give an intuitive understanding via an example as well as an algorithmic description.

Example 5. *Let $G = (V, E, w)$ be a fuzzy graph with $w(e_{ij})$ denoting the membership strength of the edge between nodes $v_i, v_j \in V$. Furthermore, assume that all nodes belong to the classes C_1 or C_2, but we only know the classes of a subset of the nodes.*

To propagate the label information, we use belief propagation. For every node v_i with unknown class, we assume the prior $P(v_i = C_1) = 0.5$. We then perform a step of

belief propagation by using Bayes' theorem as

$$P(v_i = C_1|V) = \frac{P(v_i = C_1)\sum_j \left(w(e_{ij})P(v_j = C_1)\right)}{\sum_j w(e_{ij})} \tag{2.18}$$

where $P(v_i = C_1|V)$ should be interpreted as the probability of v_i having class C_1, given information about the classes of all neighbouring points.

2.4.1.1 Algorithmic description

In the general case with c separate classes, belief propagation can be performed efficiently using simple matrix operations. Let $\Gamma \in R^{N\times N}$ be the adjacency matrix representing the fuzzy graph $\tilde{G}_z$, built on a k-NNG as described in Section 2.3.4. The adjacency matrix is sparse, at most $2k$ elements per row are nonzero. All its elements are in the interval $[0, 1]$, with elements representing the probability that an edge is present. Furthermore, let $L \in R^{N\times c}$ represent the label matrix, with rows corresponding to data samples, and columns to class probabilities. Given Γ and L, we can define the belief propagation algorithm as described in Algorithm 1. The stopping criterion can either be a fixed number of iterations that should be performed or something that measures how much the labels have changed in recent iterations.

Algorithm 1 Label Propagation

1: Initialise L and Γ
2: **while** stopping criterion not met **do**
3: $\quad \tilde{L} \leftarrow L \odot \Gamma L$
4: $\quad L \leftarrow \dfrac{\tilde{L}}{\sum_j \tilde{L}_{ij}}$
5: **end while**

3 Methods

The first feature set Z, with elements z, hereafter referred to as the *latent vectors*, is acquired by training an autoencoder to reconstruct spectrograms of the waveforms, and the second set U, with elements u, hereafter referred to as the *embeddings*, is obtained by projecting the first set with UMAP. This decreases both the dimensionality further and also allows for the enforcement of invariance properties. An overview of the complete feature pipeline is depicted in Figure 3.1.

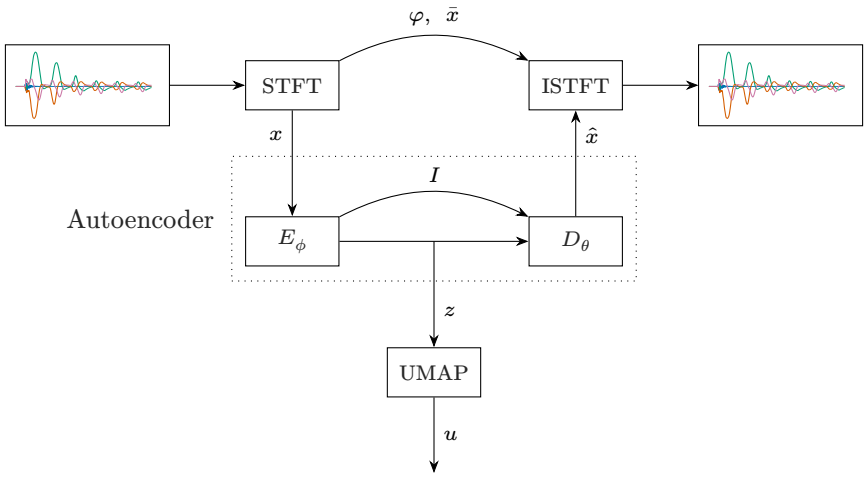

Figure 3.1: Overview of all components necessary to produce the two feature sets z, and u. As z is desired to have certain invariance properties, phase information φ and steady state behaviour $\bar{x}$ is deliberately removed when calculating the spectrogram. Consequently, to reconstruct the complete waveform it is necessary to use φ and $\bar{x}$. Likewise, the max-pooling indices for each channel are given by the encoders to the decoders to reconstruct the spectrogram.

3.1 Data preprocessing

The autoencoder model does not operate directly on the raw waveforms. Instead, the current and voltage waveforms are normalised separately and then converted into spectrograms using the Short-time Fourier Transform.

3.1.1 Normalisation

Common normalisation techniques include min/max-normalisation and z-normalisation defined as

$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)} \quad \text{and} \quad \tilde{x} = \frac{x - \mu}{\sigma}$$

respectively, where μ and σ denote mean and standard deviation. These scalings can be applied on whole datasets, batches, samples or individual features. Although more normalisation often leads to better numerical stability, it is important to consider what information is lost during normalisation.

In the case of data from three-phase power systems, both current and voltage waveforms already have mean close to zero. In addition, the relationships within the three currents and voltages contain crucial information. Subtracting anything from the measurements might therefore be unnecessary, and normalising channel-wise could lead to the loss of inter-channel relationship. Normalisation is therefore performed as,

$$\tilde{x} = \begin{bmatrix} c_1 \cdot x_{\text{currents}} \\ c_2 \cdot x_{\text{voltages}} \end{bmatrix} \tag{3.1}$$

where c_1 and c_2 are constants. In this thesis, they are chosen to be the median of the reciprocal of the Root Mean Square (RMS) values across the whole dataset. The root mean square values can be calculated as

$$\text{RMS} \doteq \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{3.2}$$

where N is the number of samples in the signal snippets. Taking the median of all RMS values is reasonable if the majority of the dataset contains mostly non-anomalous recordings. Otherwise, the same procedure can be repeated on a smaller part of the dataset which contains mostly normal recordings.

3.1.2 Short-time Fourier transform

The discrete short-time Fourier transform is used to transform each channel (current and voltage phases and neutral) of the signal into the time-frequency domain. Applying STFT and then taking the absolute value to discard the phase information results in a spectrogram.

When constructing spectrograms, there is a trade-off between time and frequency resolution, which is determined by a window size. In this case, the lowest frequency of interest is the fundamental frequency. The window size k is therefore chosen to be the same as one period of the fundamental frequency found in the current and voltage waveforms, i.e. 80 for data sampled at 4000 Hz since the grid frequency is 50 Hz. It is not necessary to apply a windowing function, as is common when applying the STFT to audio signals since the fundamental frequency is known and spectral leakage is therefore not a concern. This choice of parameters allows for an easy interpretation of the spectrogram: The first row corresponds to the DC frequency, whereas the consecutive rows correspond to the base frequency and its harmonics.

Given this choice of parameters, a simple version of the STFT of an input signal x can be defined as

$$\text{STFT}\{x\}(\tau, \omega) \doteq \text{FFT}\{x[\tau k : \tau k + k]\}(\omega)$$

where k is the window size, equal to the number of samples in one period. The parameter ω takes values $(0, 1, \ldots, k/2)$ according to the Nyquist-Shannon sampling theorem [34], and the parameter τ takes values $(0, 1, \ldots, 50t - 1)$, where t denotes the length of the recording in seconds.

It is assumed that the interesting part of the recordings is when something changes in the system. To obtain a spectrogram representation reflecting this, we subtract the average frequency content during normal operation from the spectrogram. Many PQ monitors, including the ones used in datasets A and B, save recordings that have some margin at the start, before the disturbance happens. This makes the first part suitable for calculating the average frequency content during normal operation.

An example spectrogram, both with and without subtracted medians, can be seen in Figure 3.2. After the recordings are normalised and transformed to magnitude

spectrograms with subtracted mean frequencies, they are ready to be fed into the autoencoder for feature extraction.

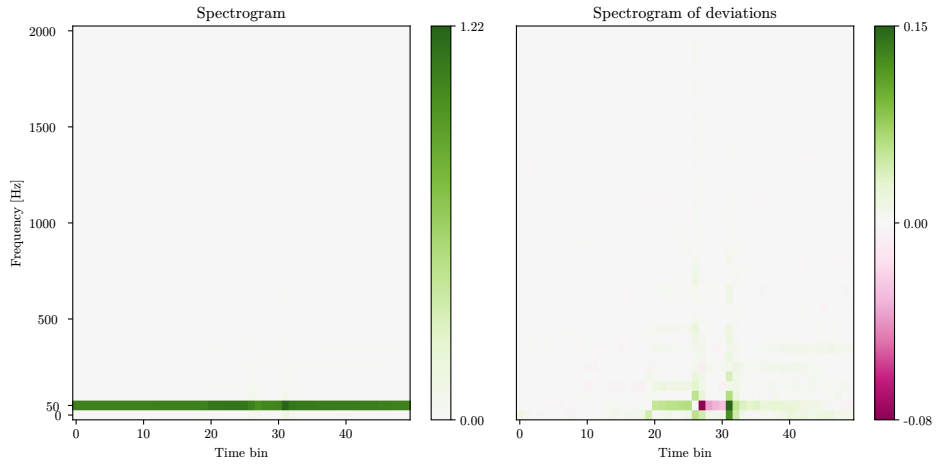

Figure 3.2: Spectrograms of phase L1 of an example disturbance. The left spectrogram is a normal spectrogram and the fundamental frequency makes it hard to distinguish other frequency content. The right spectrogram has subtracted medians per frequency bin, which is what the autoencoder receives as input.

3.2 Autoencoder architecture

The autoencoder consists of four encoders and four decoders, which have the same architecture but different trainable parameters. The input channels are organised into four groups, namely phase currents I_{L1}, I_{L2}, I_{L3}, phase voltages U_{L1}, U_{L2}, U_{L3}, neutral voltage U_N and neutral current I_N. Each channel is processed separately through its corresponding encoder and decoder. For each channel, a collection of c latent variables is saved. These variables are then concatenated into a latent vector of size $n_z = 8c$. The purpose of this is to achieve a permutation equivariant network, meaning if we change the order of two phases, the corresponding permutation is also achieved in the latent vector. For a high-level view of the architecture, see Figure 3.3.

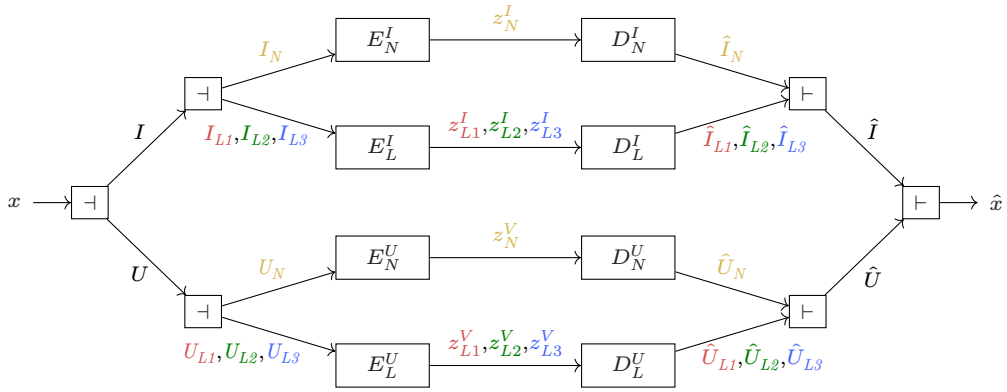

Figure 3.3: Block diagram of the autoencoder architecture, where the spectrogram input is split into current and voltage and then further into neutral and phases. These components undergo processing by their respective encoders E and decoders D. Using this architecture, equivariance in the latent space z with respect to phase permutations is obtained.

Each encoder and decoder is largely made up of convolutional layers, which are

commonly used when extracting time-invariant features. Readers unfamiliar with this concept are advised to view [27] for a concise introduction. Notably, only *causal* convolutions are used here, i.e. the output in layer ℓ at time bin t only depends on time bins t and earlier in layer $\ell - 1$. This variation has been used extensively in the literature. In order to make the latent space completely invariant with respect to translation in time, as well as sample length, an adaptive max pooling layer is used which preserves the maximum indices in each channel while letting the latent features be the maximum values. Additionally, the ReLU function is added after the max-pooling operation. For the decoder to properly reconstruct the signal, the indices saved in the max pooling layer in the encoder are forwarded to an unpooling layer in the decoder. There are two hyperparameters in the encoder and decoder, namely the kernel size k and the number of kernels c. While c directly impacts the number of latent variables, k only affects how large the receptive field is, see Figure 3.4. In conclusion, the autoencoder extracts a feature vector z which is *equivariant* with respect to phase permutations and *invariant* with respect to translation in time as well as sample length.

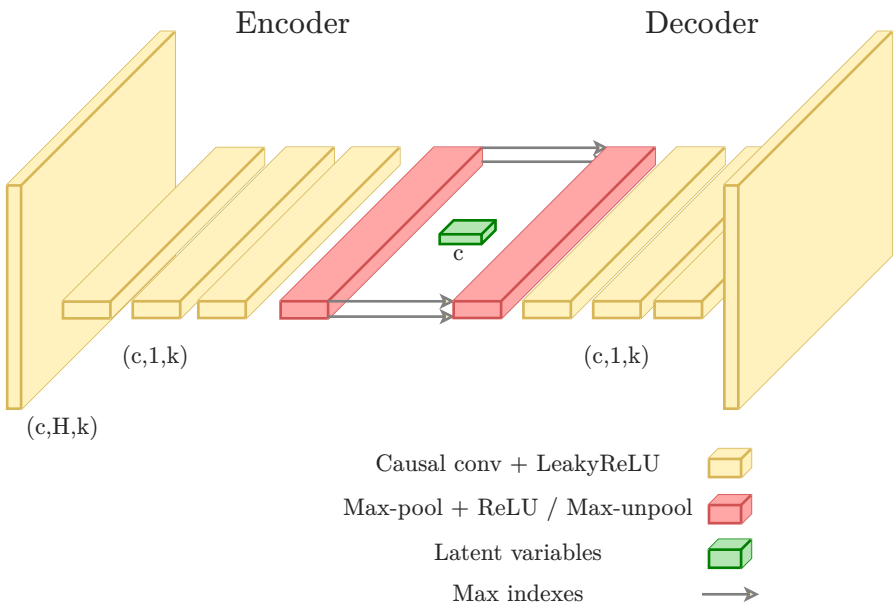

Figure 3.4: An encoder-decoder pair that takes a spectrogram as input. The numbers c and k are hyperparameters while H denotes the height of the input spectrogram, i.e the number of frequency bins. In other words, the first convolutional layer collapses all the frequencies and the last convolutional layer unpacks them back to a spectrogram.

3.3 Permutation invariant embedding

In order to embed the autoencoder features into a lower dimension using UMAP, it still remains to choose a pseudo-distance d. For PQ data, the naming of the phases $L1$, $L2$, and $L3$ is arbitrary, and for some purposes, such as fault classification, a representation of the dataset that is invariant to phase permutations is ideal. For three-phase power systems, there are three possible cyclic permutations, namely $[L1, L2, L3]$, $[L2, L3, L1]$ and $[L3, L1, L2]$. Electricity is (for the most part) generated by synchronous machines that have a fixed direction of rotation; therefore, applying antisymmetric permutations such as $[L1, L3, L2]$ is not representative of real data. The autoencoder architecture already grants equivariance with respect to phase permutations, and by using UMAP together with the custom pseudometric d, the dimensionality of the latent vector z can be further reduced while gaining invariance

with respect to phase permutations. The dimensionality of the embedding can either be chosen to be 2, which is optimal for visualisation purposes, or something larger, which could preserve more of the local structure of the dataset.

Given the feature set extracted from the autoencoder Z and a distance function ℓ acting on elements $z_i \in Z$, let σ be the operator that performs cyclic permutations of the phases in z_i. Note that this is only possible to do since the latent vector z is distinctly separated into parts corresponding to specific phases and the neutral channel. If all eight channels of the spectrogram had been processed by a single convolutional encoder, there would be no clear separation.

A pseudometric $d : Z \times Z \mapsto \mathbb{R}_{\geq 0}$ can be defined as

$$d(z_1, z_2) \doteq \min \left\{ \ell(z_1, z_2), \ell(z_1, \sigma z_2), \ell(z_1, \sigma^2 z_2) \right\}. \tag{3.3}$$

In other words, d is defined to be the smallest possible distance obtained by phase permutations. d is not a proper distance, as multiple points could have 0 distance to each other. However, in Appendix A.1 it is shown that d is a pseudometric. In our results, we have chosen ℓ to be Euclidean (L2) distance, although Manhattan distance (L1) has given similar results.

4 Results

In this section we present results regarding informativeness, compactness and interpretability of the autoencoder features z and UMAP features u. As u depends on z, a natural first step is to determine an appropriate size for z, which is done in Section 4.1. Thereafter, the possibility of reconstructing the original waveforms from the reconstructed spectrogram is investigated in Section 4.2. It is also explored how changing the latent space representation influences the reconstruction in Section 4.3. This is followed by an examination of how z can be made sparser in Section 4.4. In Section 4.5, the potential of using transfer learning for the two datasets A and B is explored. Both z and the benchmark manual features z_m described in section 1.5 are then projected into two dimensions using UMAP for visualisation purposes in Section 4.6. The classification downstream task is evaluated in Section 4.7.1 where a similar classifier is trained on all feature sets z, u and z_m, plus an additional benchmark feature set, calculated with PCA and FFT. As an alternative to classification with neural networks, Section 4.7.2 presents results when the belief propagation method described in Section 2.4.1 is used. Section 4.8 presents unsupervised clustering on z using HDBSCAN, followed by the final downstream task evaluation of predictive fault risk estimation in Section 4.9.

4.1 Reconstruction error and latent size

The reconstruction loss $||x - \hat{x}||$ for an autoencoder measures how similar the reconstruction $\hat{x}$ is to the original signal x. Intuitively, if a decoder is able to reconstruct an original signal with low reconstruction error from a set of features, then the latent features must contain considerable information about the original input. Another important aspect is the dimensionality of the features. Smaller features are naturally more desirable but they also generally incur a higher reconstruction loss and so a compromise must be made.

The latent vectors of the autoencoder have size $n_z = 8c$, where c is a hyperparameter, reflecting the fact that there are 8 channels in the input data. Recall that this dataset contains signals with 4000 samples resulting in size 8×4000. To determine a suitable size of the latent space, the autoencoder is trained to minimise the spectrogram reconstruction error for several different latent sizes on dataset A. Of the total data, 20% is used as validation data, with the remainder used for training. The L1 reconstruction loss as a function of training epoch for the different latent sizes is shown in Figure 4.1. Further insight can be gained by investigating how the reconstruction error is distributed across samples. In Figure 4.2, a scatter plot of the reconstruction error $||\hat{x} - x||_1$ is plotted against the spectrogram norm $||x||_1$, for three different latent sizes. The dotted line represents cases where the reconstruction is filled with zeroes, and it is clear that larger latent sizes allow for more non-trivial reconstructions, especially for large $||x||_1$.

Judging by both Figure 4.1 and Figure 4.2, it seems like a larger latent size implies a lower reconstruction error for validation data. However, as the manually selected features z_m have dimensionality 149, and we want to make comparisons between z and z_m, it is deemed most fair to let the latent size be 128 for the remaining results.

4.2 Waveform reconstruction

Although the reconstruction loss might be low for the spectrograms, it can be difficult to understand how this relates to the original waveforms. To examine this, the

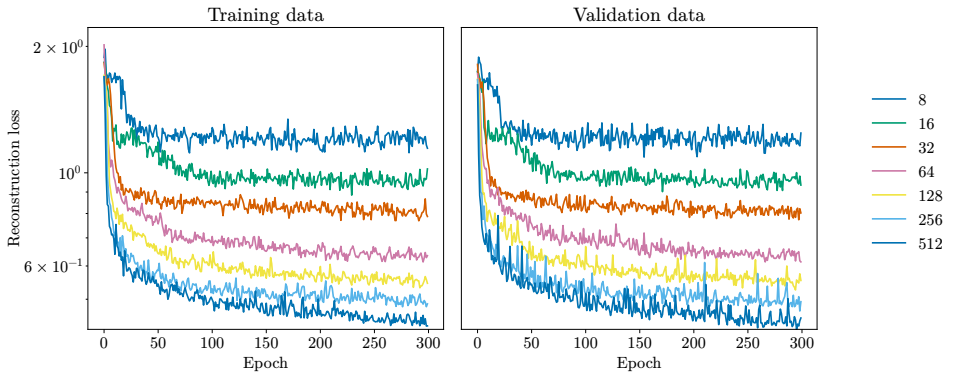

Figure 4.1: Reconstruction loss on training and validation data from dataset A across epochs for different latent sizes n_z.

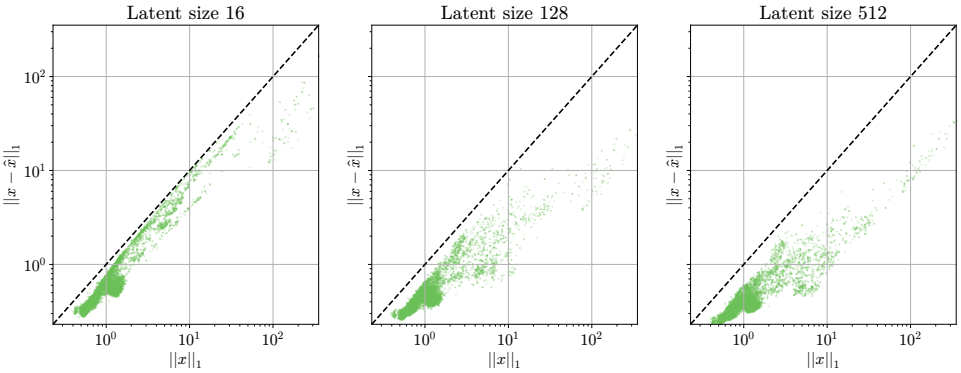

Figure 4.2: Scatter plots of L1 reconstruction error $||x - \hat{x}||_1$ vs L1 norm of the processed spectrogram $||x||_1$.

inverse STFT is taken from the reconstructed spectrograms by using the same phase information that was discarded when performing the STFT, see Figure 3.1. Original recordings and reconstructions for two representative signals from the validation set of dataset A are shown in Figure 4.3 and Figure 4.4. It is clear that the autoencoder preserves the main characteristics of the disturbance, even though there are still some deviations.

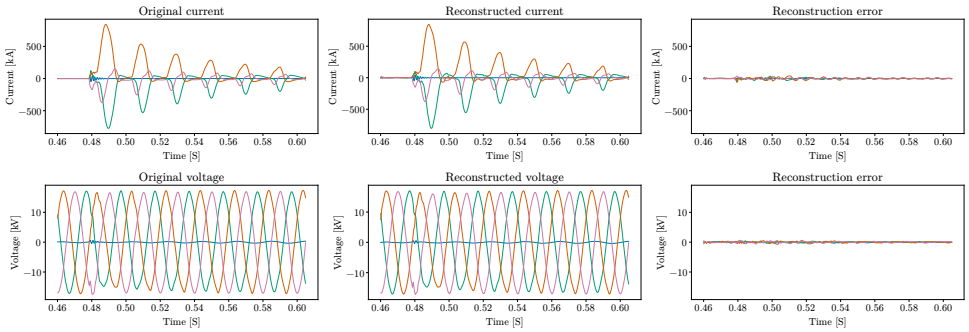

Figure 4.3: A recording to the left, its reconstruction in the middle, and the difference to the right.

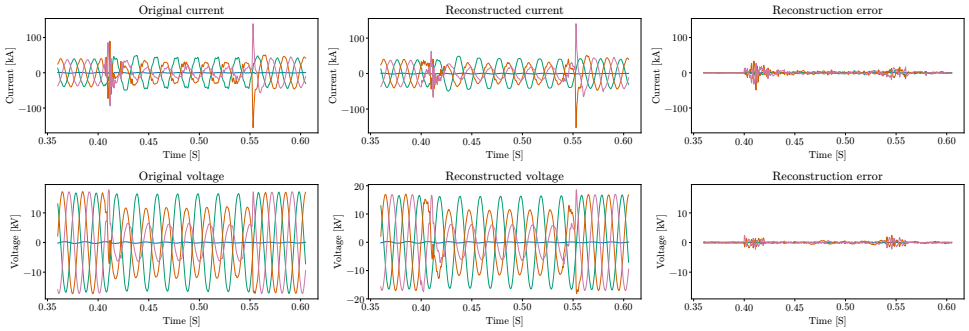

Figure 4.4: A recording to the left, its reconstruction in the middle, and the difference to the right. The reconstruction is not perfect, for example, the reconstruction has a smaller current right as the disturbance begins.

4.3 Manipulating the latent space

The decoder can reconstruct recordings with high accuracy, confirming that the latent space contains most of the information about the disturbance. Depending on what the features are used for, the interpretability of the latent vector might be of high importance. One way to achieve this is to perturb specific elements of the latent vector or the corresponding indices and then observe how the reconstruction changes. For each new autoencoder that is trained, the elements of the latent vectors carry different information; thus, exploration of one specific latent space will not necessarily generalise to other models. However, the example shown in Figure 4.5 is very representative; small changes always result in well-behaved changes. Furthermore, increasing the magnitude of the latent elements increases the severity of the disturbance and shifting the indices shifts the corresponding disturbance. It is also seen that the different phases and neutral are decoupled: only changing a single latent element will only affect one of the phases or neutral.

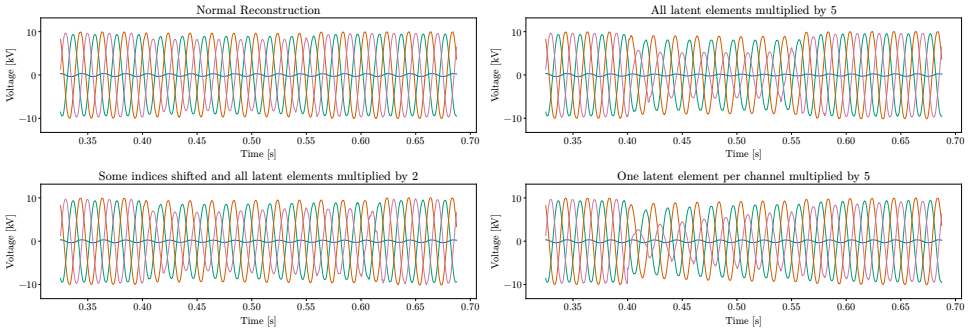

Figure 4.5: On the top left, a reconstruction is shown. On the top right, the latent vector z is multiplied by a constant before being passed to the decoder. On the bottom left, some of the indices are shifted to the right before being passed to the decoder. Finally, on the bottom right, a specific element of the latent vector z is perturbed, showing a change of character of the reconstructed disturbance.

4.4 Enforcing sparsity in the latent space

Having sparser latent vectors, i.e. latent vectors with more zeros, would arguably increase the interpretability of the features. For example, it would be ideal if a recording where nothing of interest happened had a corresponding latent vector with only zeros in it. As described in Section 3.2, a ReLU is applied to the latent vector z before passing it to the decoder, which means that multiple elements in

the latent vector are likely zero. Intuitively, one would expect that the recordings with the smallest disturbances have the most zeros, while recordings of more violent disturbances would result in a latent vector with fewer zeros since the encoder needs to convey more information to the decoder. As seen in Figure 4.6, there does indeed seem to be some correlation between the two. However, the number of nonzero elements is never close to zero.

To see if sparsity can be achieved without retraining the autoencoder, an element-wise, modified ReLU function is defined as

$$f_\varepsilon(x) = \begin{cases} 0 & \text{if } x < \varepsilon \\ x & \text{otherwise} \end{cases}$$

where $\varepsilon \geq 0$ is a parameter that enforces sparsity. Note that if $\varepsilon = 0$, the function f is simply the ReLU function and does nothing when applied to z. To find a reasonable value for ε, the *sparse reconstruction*

$$\hat{x}_\varepsilon = D(f_\varepsilon(E(x)))$$

and the corresponding *sparse reconstruction error*

$$||\hat{x}_\varepsilon - x||_1$$

is defined. To see how ε influences reconstruction and sparsity, a random subset of dataset A is selected and the sparse reconstruction error is plotted against the number of nonzero elements in z_ε, see Figure 4.6. It can be concluded that setting ε such that 50 % of the latent variables are zero barely affects the reconstruction error. For simplicity's sake, the remaining results are presented without sparsity, although including sparsity would likely yield similar results.

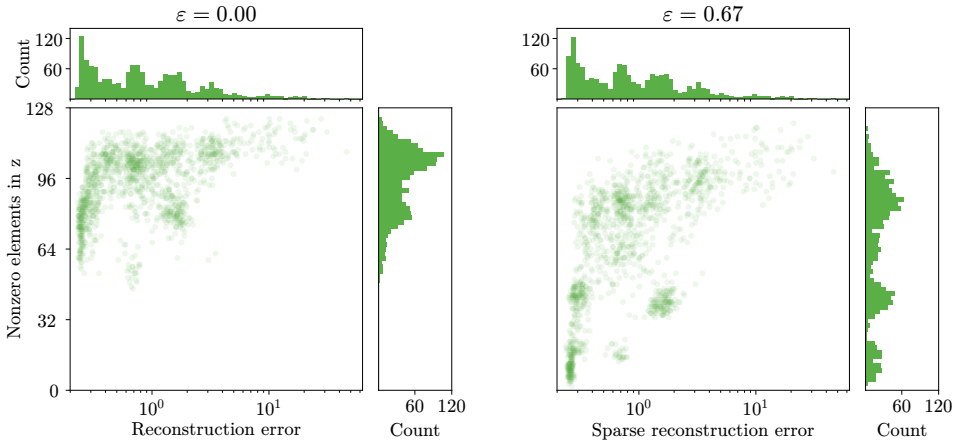

Figure 4.6: Scatter plots showing how enforcing sparsity in the latent vectors affects reconstruction. To the left, $\varepsilon = 0$ while in the right plot ε is chosen so that on average 50 % of the latent variables are zeroed out.

4.5 Transfer learning

When new PQ monitors are installed in an existing substation, using a pretrained autoencoder to extract features from new recordings is clearly more convenient than waiting until a sufficient amount of data is collected. To facilitate transfer learning, the data needs to share sample rate and have similar magnitude. Resampling is necessary so that every row in the spectrogram representation corresponds to the same physical frequency across datasets. Varying recording lengths do however not pose any problems for the autoencoder architecture. There are multiple ways to resample two sets of signals to a common sample rate; however, downsampling is

easier than upsampling. To avoid aliasing, downsampling is performed by cropping the signal in the Fourier domain, saving only the low-frequency part, and then taking the inverse Fourier transform.

To evaluate how transfer learning between dataset A and dataset B improves reconstruction quality compared to training an autoencoder only on dataset B, two models are first randomly initialised. The first autoencoder is then trained on dataset A, followed by fine-tuning on dataset B, while the second is only trained on dataset B. An additional third model is also introduced that pretrains on speech data from LibriSpeech [30], a dataset that contains 100 hours of clean voice recordings for audiobooks, with a sample rate of 16 kHz. The dataset is partitioned such that each sample contains 0.25 s of a voice recording, and the same window size of 80 is used as for the other datasets, which means that the frequency resolution is 100 Hz. A typical spectrogram sample from this dataset is seen in Figure 4.7, which should be compared to the PQ spectrograms in fig. 3.2. A major difference with the audiobook spectrograms is that they are often 0, even before the medians for each frequency are subtracted since speech often contains pauses. However, after subtracting the medians, this discrepancy disappears. Another differing aspect of the audiobook spectrograms is that the disturbances in them are often spread over a larger variety of frequencies, as opposed to PQ spectrograms that mainly have their disturbances concentrated around the grid frequency.

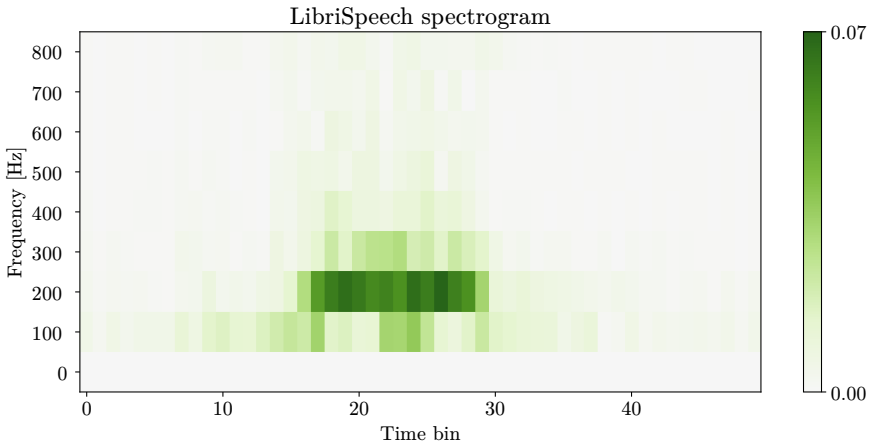

Figure 4.7: Spectrogram of a 0.25 s sample from the LibriSpeech dataset, with a sample rate of 16 kHz and window size 80.

Dataset B is split into half for a train set and a validation set, with the same validation set shared between all models. The two pretrained models are allowed to pretrain for 200 and 20 epochs respectively, before switching over to train on the training half of dataset *B*. This procedure is repeated 8 times to obtain an estimate of the variance of each model. In Figure 4.8, the model with lowest final validation loss of each model is plotted in a solid line, while the model with median final validation loss is plotted with a transparent line. Both pretrained models clearly achieve a lower validation loss and the model trained on the LibriSpeech dataset performs the best, perhaps surprisingly. One possible conclusion from this result is that the size and variety of the pretraining dataset is more important than the likeness to the dataset used for fine-tuning.

For a view of the relative reconstruction error distribution, see the histogram in Figure 4.9, where the best models from Figure 4.8 have been used. Notably, the model pretrained on the LibriSpeech dataset manages to reconstruct a significant amount of samples with lower relative reconstruction error than 0.3, while the randomly initialised model has almost no such samples.

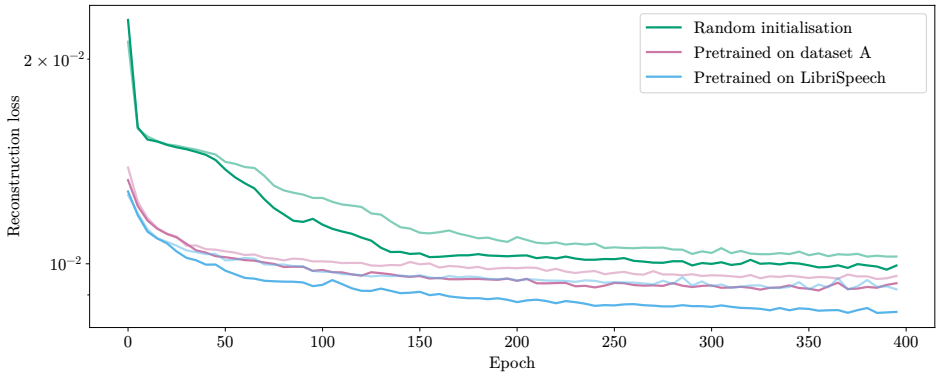

Figure 4.8: Epoch validation loss on dataset B for a model trained only directly on dataset B, a model that is also pretrained on dataset A, and a model that is pretrained on the LibriSpeech dataset. Solid lines show the epoch loss trajectory for the models with the lowest final validation loss while the transparent lines represent the trajectories with median final validation loss.

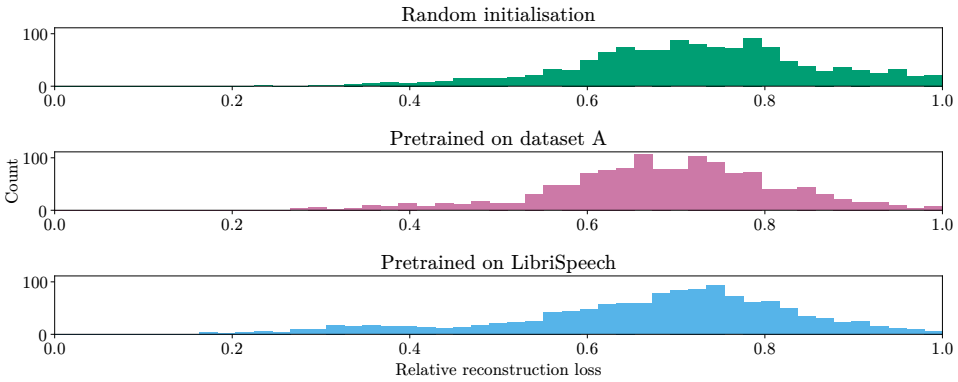

Figure 4.9: Histogram of relative reconstruction error for both pretrained models and the model trained only on dataset B.

4.6 UMAP VISUALISATION

Visualising a set of features can have several use cases. It can be used to manually specify clusters, for data exploration, and for labelling, where the user can quickly inspect if all samples in a given cluster have a similar label. Using feature vectors $z \in \mathbb{R}^{128}$ and manual features $z_m \in \mathbb{R}^{149}$, which both correspond to a labelled subset of dataset A, we can visualise the two feature sets using UMAP in two dimensions. They are plotted beside each other in Figure 4.10. Recall that the pseudometric used to construct the embedding is invariant with respect to phase permutations; thus, the clusters will not depend on which phases are affected by the PQ event. For both feature sets, some clustering occurs, and it seems that the clusters roughly correspond to the classes, even though the class information has not been used during the projection. For the manual features, samples within the same class seem to be placed in different clusters more often than compared to the autoencoder features.

To examine whether the position within a cluster reflects significant signal changes in the embedding, we visualise the signals for points within two distinct clusters in Figure 4.11. As the points approach the blue cluster, which represents non-anomalous signals, the disturbances in the signals gradually diminish.

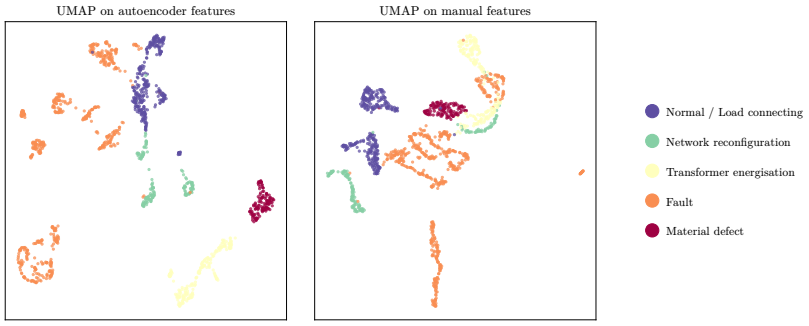

Figure 4.10: 2D embeddings based on two different feature sets; extracted with an autoencoder and features chosen by a domain expert.

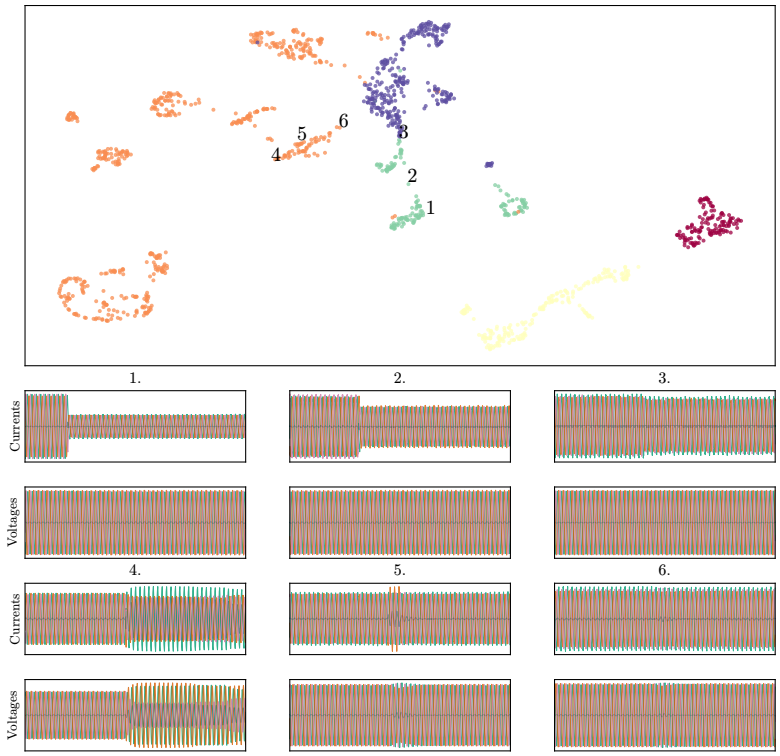

Figure 4.11: 2D UMAP embedding of autoencoder features z, using the permutation invariant pseudometric. Six points are chosen from two different clusters, and their corresponding waveforms are shown below the scatter plot.

4.7 Classification

One of the primary downstream tasks based on the extracted features is the classification of root causes. We therefore compare both the autoencoder features and UMAP features with two other baseline features in terms of classification perfor-

mance. For a brief explanation of the macro F_1-score, which is used as a metric, see appendix A.2.

4.7.1 Classification using neural networks

One way to evaluate classification performance is to train a similar neural network classifier on all feature sets, for the labelled subset of dataset A described in Section 1.4.1, and classes described in Table 1.2, with 50% of the labelled data used for validation. The autoencoder features still have dimension $n_z = 128$ and the UMAP features are chosen to have size $n_u = 5$. The two baseline features are the manual features described in section 1.5, and additionally, features calculated with Principal Component Analysis (PCA) on the concatenated, z-normalised, channel-wise FFT of the raw data. This set of features is primarily included as a benchmark for what an initial, naive feature set might look like. In Figure 4.12, the validation loss and macro F_1-score for a classifier trained on each set of features is shown. The classifier itself is a fully connected neural network with one hidden layer of size 128. Evidently, all feature sets except the PCA ones seem to be converging to a similar macro F_1-score, although the manual features perform slightly worse at 96.8% compared to UMAP features 97.9% and autoencoder features 98.4%. Curiously, the autoencoder features perform better than the UMAP features, although they are not phase-permutation invariant. We suspect that the primary cause for this is simply the larger dimensionality. The validation loss for the manual features exhibits peculiar behaviour, and also starts at a higher value, with the cause seemingly being a lack of normalisation. Normalising the features using min-max scaling or z-normalisation makes the validation loss smoother, but leads to a substantial decrease in classification performance, and is not shown here.

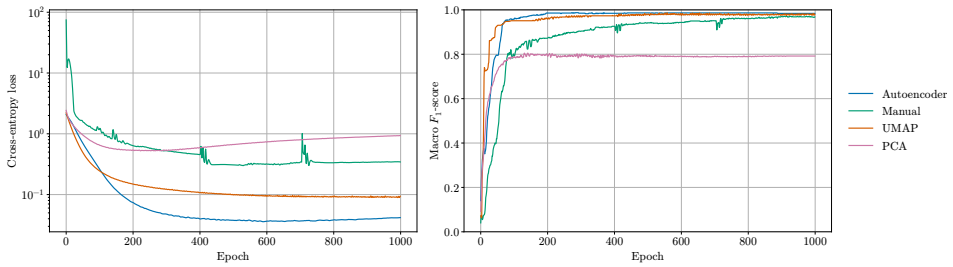

Figure 4.12: Validation loss and macro F_1-score for a neural network classifier trained on the four different feature sets, calculated from dataset A.

Although the feature sets (ignoring PCA) have similar classification performance, it is interesting to see which classes each classifier struggles to identify. Therefore, in Figure 4.13, the confusion matrices for each classifier are presented. Recall from Table 1.2 that the classes are sorted according to severity level related to power quality. Therefore, elements in the confusion matrices far from the diagonal are undesirable, since it means that the classifier has confused two classes with different severity levels. From this perspective, the classifier trained on UMAP features can be said to perform the best, since it confuses the least severe class with two other non-severe classes, while the classifiers trained on the other feature sets confuse the second-most severe class with other less severe cases.

We have argued that phase-permutation invariance is one of the two primary advantages of the UMAP features. To see whether this is the case for classification, the same experiments are repeated but with the validation data phase-permuted clockwise. The mean and standard deviations of the final macro F_1-scores across 10 training runs for both the original validation data and the permuted validation data are presented in Table 4.1. The UMAP features perform similarly on both validation sets, while the two benchmark features perform substantially worse. Interestingly, the autoencoder features also perform very well on phase-permuted validation

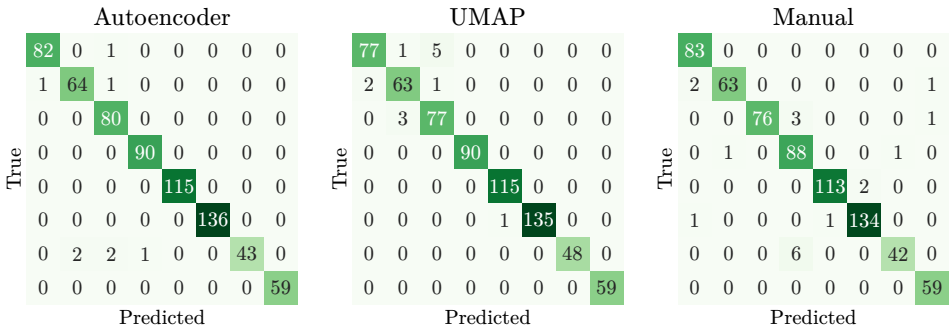

Figure 4.13: Confusion matrices for classifiers trained on the autoencoder, UMAP, and manual features.

data, and we suspect that this is due to the neutral channel still containing enough information to allow classification. To test this hypothesis, we also evaluated the autoencoder features without the neutral part. As suspected, those features perform much worse. Note that this result is quite relevant since there might be scenarios where the recorded waveforms do not include a neutral channel. In conclusion, both of our feature sets perform similarly on the classification task but since the UMAP features are significantly smaller and not as dependent on the presence of the neutral channel, they should be preferred over the autoencoder features.

Table 4.1: Mean and standard deviations, estimated across 10 training runs, of the macro F_1-scores for each model. The columns denoted original and permuted present performance on the original validation set as well as the same validation set but phase-permuted clockwise.

Features	Dim.	Validation F_{m}-score	
		Original	Permuted
Autoencoder	128	**0.982±0.002**	**0.984±0.002**
UMAP	5	**0.981±0.001**	**0.983±0.002**
Manual	149	0.969±0.006	0.849±0.030
PCA	128	0.785±0.011	0.610±0.018
Autoencoder without neutral	96	**0.975±0.004**	0.834±0.013
UMAP without neutral	5	0.950±0.006	**0.950±0.006**

One advantage of having smaller feature vectors is that models trained on them often require fewer training samples and are less prone to overfitting. Therefore, one would expect a classifier trained on the UMAP features to perform better for low amounts of training data. For the next result, different training size ratios of dataset A are used, ranging from 50 % (as previously) to only 8 samples, or one sample per class. Since the number of examples in each epoch will vary drastically, instead of using a fixed number of epochs for training, the training is stopped when the validation loss has not decreased for 100 epochs. In Figure 4.14 it is seen that both the autoencoder features and UMAP features perform better than the manual ones, but that the UMAP features perform best of all for small training sets.

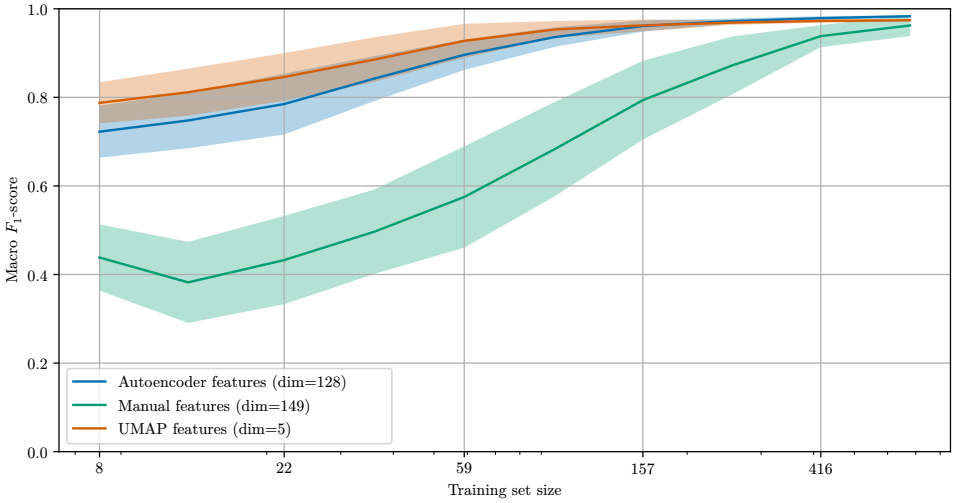

Figure 4.14: Macro F_1-score as a function of training set size. The learnt features generalise significantly better for smaller training sets.

4.7.2 Classification using label propagation

We have just shown that UMAP features are suitable for classification, however, to obtain the UMAP features, we have to choose an embedding dimension and then minimise the cross-entropy between graphs as described in Equation (2.10). The embedding dimension has to be somewhat arbitrarily chosen, and the cross-entropy between the fuzzy graphs will never be zero in practice, meaning that we lose some information in the optimisation step. Motivated by this, we try to perform classification directly on the fuzzy graph created using methods from Section 2.3.4.1. Using label propagation as described in Section 2.4.1, a few labels are extended to the whole dataset. The macro F_1-score as a function of training set size (labels which are propagated) is shown in Figure 4.15. Apparently, using label propagation is a better method than applying our fully connected classifier for this classification task.

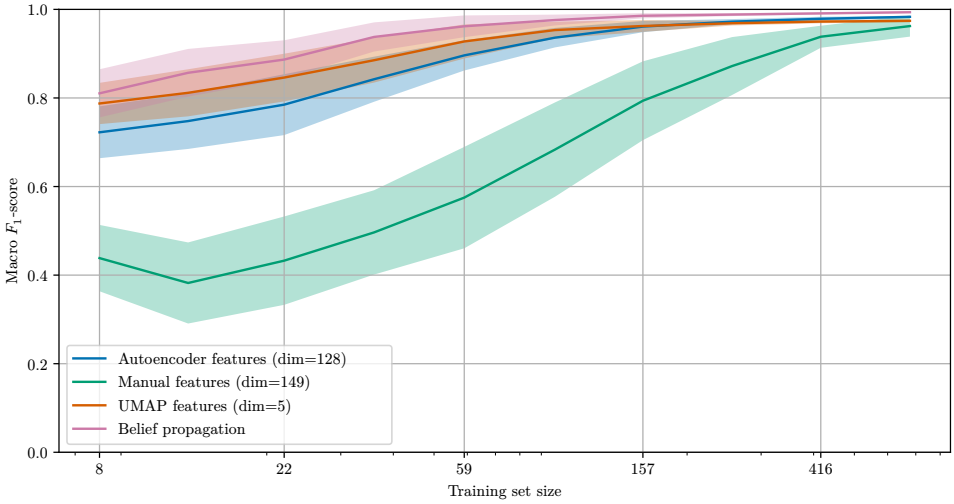

Figure 4.15: Macro F_1-score as a function of training set size. The classification performed by label propagation is the most accurate for every training set size.

The mean and standard deviation F_m-scores obtained when using the same training

and validation dataset as in Table 4.1 is 0.994 ± 0.002, which is significantly higher than any of the results obtained by using a neural network classifier.

4.8 Clustering

Unsupervised clustering is a complex problem in high-dimensional space. The embeddings U have low dimensionality, making them suitable for applying algorithms such as HDBSCAN [24]. An example of unsupervised clustering applied to U is shown in Figure 4.16.

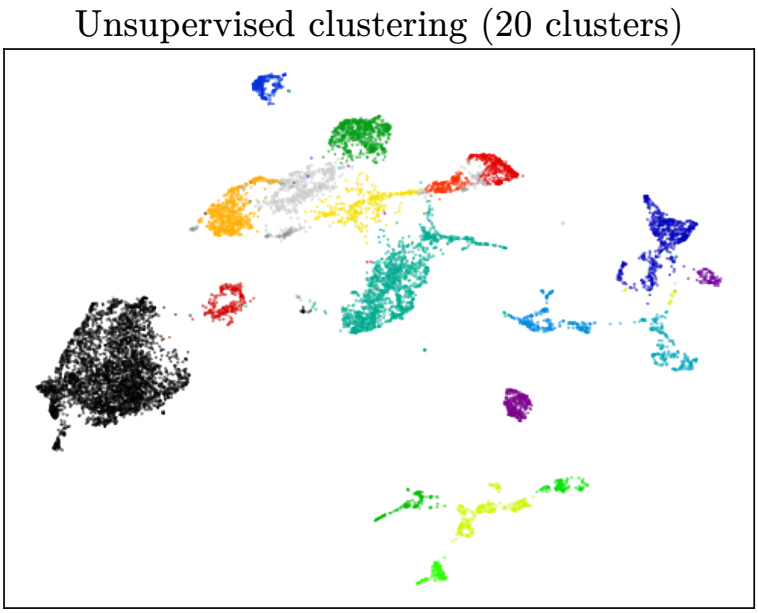

Figure 4.16: Scatter plot representing dataset A, where colours indicate clusters that have been obtained by applying HDBSCAN to the embedding.

Clusters can be used as a starting point for creating labels, or as input to other downstream tasks such as fault risk estimation. To see how well the clusters correspond to the labels used in Section 4.7, we separate dataset A into clusters, but then select only the subset of points that have labels. The cluster-to-class correspondence is shown in Figure 4.17, where UMAP and HDBSCAN have been applied to both the manual features Z_m and the autoencoder-based features Z. The results indicate that clusters based on Z resemble the classes much more. For this experiment, the UMAP parameters used for both projections were Euclidean metric, nearest neighbour $k = 30$ and *min-dist* $= 0$. Note that some clusters do not contain labelled points and that the corresponding clusters are excluded from Figure 4.17. In general, smaller feature sets are needed when data is scarce. As the total number of recorded faults might be few, and the number of recorded and *predictable* faults might be even fewer, using only cluster information for fault risk estimation might be a good starting point.

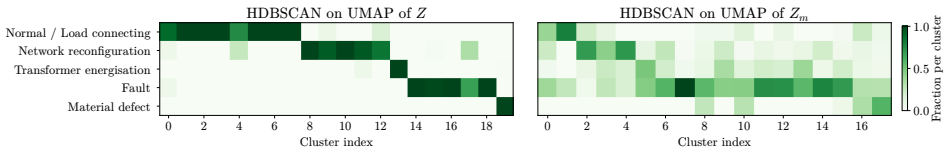

Figure 4.17: Correspondence between labels and unsupervised clusters obtained using HDBSCAN on two feature sets. It is clear that the feature set Z gives clusters which correspond more closely to the labels.

4.9 Predictive fault risk estimation

The final downstream task we evaluate our features on is predictive fault risk estimation. The proprietary Long Short-Term Memory (LSTM) forecasting model described in [1] is trained on dataset A, with both its original input z_m, and the cluster indices $f(u)$, with the exact form of the clustering not presented here for confidentiality reasons. For more details about the model architecture and how it is trained, refer to the original paper. The model outputs a probability for a fault occurring during a time horizon of one week. A plot of precision and recall, and equivalently the ROC curve, for both models on validation data, trained with several different weight initialisations, is seen in Figure 4.18. See appendix A.2 for an explanation of how to interpret such plots. The predictions are far from perfect, which is expected as some disturbances are unpredictable. However, the embedding appears to hold more useful information for forecasting, as is also seen in Table 4.2.

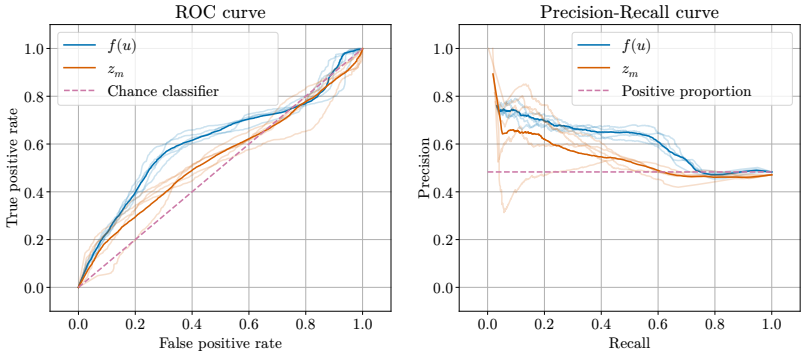

Figure 4.18: ROC and precision-recall curves for a fault risk estimation model trained on the manual features z_m and a processed version of the embedding $f(u)$. The opaque lines are averages of the transparent lines, which represent different weight initialisations.

Table 4.2: AUROC and AUPRC scores for u and z_m respectively.

Metric	u	z_m
AUROC	0.60	0.54
AUPRC	0.59	0.53

5 Conclusion

In this thesis, we present a multi-step method for extracting two types of features from three-phase waveform data. The first feature set is achieved by training an autoencoder with a convolutional architecture to reconstruct spectrograms of the waveforms, producing features that are invariant to time translation and recording length. The second feature set is a projection of the first, using UMAP with a specific pseudometric to gain both smaller and phase-permutation invariant features.

The fact that the whole method can be implemented as a parameterised model allows for additional flexibility, for example, feature quality could potentially be improved by incorporating more tasks besides reconstruction in the decoder. Furthermore, although the features obtained by the proposed feature extraction method seem to be very compact and informative, it is possible that changing certain details in the implementation might yield even better results, as no thorough exploration of hyperparameters has been conducted. Examples of such hyperparameters include kernel sizes, as well as the number of layers and channels in encoders and decoders.

Other possible modifications concern the construction of the spectrograms. The spectrogram representation used in this thesis was obtained by calculating the STFT without overlap or window function, as well as a window size of one period, following the principle of parsimony. However, it is possible that other parameter values such as a window size of two or more periods would be beneficial for certain tasks, as it is a trade-off between time and frequency resolution. Adding overlap in the spectrogram representation would reduce quantisation effects, potentially increasing the robustness of the model, at the cost of making the training of the autoencoder harder.

The autoencoder features are shown to contain enough information to enable a high-fidelity reconstruction, even when 50% of the latent variables are set to 0. Furthermore, we show that transfer learning is a viable option to decrease the reconstruction loss even more. Training on a different PQ dataset or training on audio data increases performance, and the size of the pre-training dataset seems to be a more significant factor than the exact appearance of the spectrograms. In a classification setting, both feature sets are shown to provide superior performance when training a neural network classifier on them, compared to the manual features. Furthermore, the UMAP features are demonstrated to be phase-permutation invariant when permuting the phases in the validation set. Additionally, unsupervised clustering of the UMAP features can be effectively implemented and the clusters correspond well to label information, and it is shown that the UMAP features are suitable for fault risk estimation, however, more data is likely needed to draw conclusions about the performance of such models.

Future work that could build on the results presented here, includes examining how pretraining on various datasets affects performance on downstream tasks besides reconstruction fidelity. Suitable pretraining datasets might also allow for using larger models that extract even more compact and informative features. When constructing the pseudometric used to create the UMAP embedding, some metric has to be chosen. In this report, Euclidean distance was used but it is possible that metrics such as correlation might yield embeddings with other desirable properties. On a more general note, the field of automatic feature extraction in power quality, not tuned to any particular downstream task, is a largely unexplored field where many of the advancements in deep learning have not yet been implemented. For example, it would be interesting to see if the model presented in this thesis could be made generative by using the inverse UMAP transform and determining a way to also

embed information about phase angles and indices. Furthermore, we believe that there is a lot to learn from the advancements of deep learning in audio-related tasks. However, care should be taken with applying off-the-shelf audio models directly; they would likely have to be modified, according to PQ relevance. For example, many audio models utilise a multi-scale loss where spectrograms of different window lengths are used since it is mainly the resulting *sound* of the waveform that is important to reconstruct, not the waveform itself. Another possible aspect to explore regards system identification. By modelling the underlying PQ system as a dynamic system of some order (likely quite low), it would be possible to calculate what input was expected to produce a given output, and the input then represents the features. Domain knowledge such as the dynamics of electrical transformers [9] could be useful in this scenario.

Summa summarum, automatic feature extraction using autoencoders on PQ waveform data seems to be a hitherto unexplored area. Apart from filling this gap, we have ensured that features extracted using an autoencoder can be made both time invariant and phase-permutation invariant while remaining compact, informative, and general. Using this feature extraction method, we hope to facilitate the development of methods that ensure the reliability of the power grid.

Bibliography

[1] E. Balouji et al. "Distribution network fault prediction utilising protection relay disturbance recordings and machine learning". In: *27th International Conference on Electricity Distribution (CIRED 2023)*. 27th International Conference on Electricity Distribution (CIRED 2023). Rome, Italy: Institution of Engineering and Technology, 2023, pp. 2273–2277. ISBN: 9781839538551. DOI: 10.1049/icp.2023.1220.

[2] *Big Data Access Working Group Power Quality Dataset*. July 29, 2019.

[3] M.H.J. Bollen. "What is power quality?" In: *Electric Power Systems Research* 66.1 (July 2003), pp. 5–14. ISSN: 03787796. DOI: 10.1016/S0378-7796(03)00067-1.

[4] Math H.J. Bollen et al. "Classification of Underlying Causes of Power Quality Disturbances: Deterministic versus Statistical Methods". In: *EURASIP Journal on Advances in Signal Processing* 2007.1 (Dec. 2007), p. 079747. ISSN: 1687-6180. DOI: 10.1155/2007/79747.

[5] Math HJ Bollen. *Understanding power quality problems*. Vol. 3. IEEE press New York, 2000.

[6] Jiajun Cai, Kai Zhang, and Hui Jiang. "Power Quality Disturbance Classification Based on Parallel Fusion of CNN and GRU". In: *Energies* 16.10 (Jan. 2023), p. 4029. ISSN: 1996-1073. DOI: 10.3390/en16104029.

[7] Lei Chen et al. "Power Quality Disturbance Signal Denoising Based on Overcomplete Representation". In: *IEEJ Transactions on Electrical and Electronic Engineering* 17.4 (Apr. 2022), pp. 544–555. ISSN: 1931-4973, 1931-4981. DOI: 10.1002/tee.23453.

[8] Roger Alves De Oliveira et al. "Deep Learning Method With Manual Post-Processing for Identification of Spectral Patterns of Waveform Distortion in PV Installations". In: *IEEE Transactions on Smart Grid* 12.6 (Nov. 2021), pp. 5444–5456. ISSN: 1949-3053, 1949-3061. DOI: 10.1109/TSG.2021.3107908.

[9] D. Dolinar, J. Pihler, and B. Grcar. "Dynamic model of a three-phase power transformer". In: *IEEE Transactions on Power Delivery* 8.4 (Oct. 1993), pp. 1811–1819. ISSN: 08858977. DOI: 10.1109/61.248289.

[10] Wei Dong, Charikar Moses, and Kai Li. "Efficient k-nearest neighbor graph construction for generic similarity measures". In: *Proceedings of the 20th international conference on World wide web*. WWW '11: 20th International World Wide Web Conference. Hyderabad India: ACM, Mar. 28, 2011, pp. 577–586. ISBN: 9781450306324. DOI: 10.1145/1963405.1963487.

[11] Hüseyin Erişti et al. "Automatic recognition system of underlying causes of power quality disturbances based on S-Transform and Extreme Learning Machine". In: *International Journal of Electrical Power & Energy Systems* 61 (Oct. 2014), pp. 553–562. ISSN: 01420615. DOI: 10.1016/j.ijepes.2014.04.010.

[12] European Committee for Electrotechnical Standardization. *Voltage characteristics of electricity supplied by public electricity networks*. 2010.

[13] Greg Friedman. "An elementary illustrated introduction to simplicial sets". In: (2008). DOI: 10.48550/ARXIV.0809.4221.

[14] Chenjie Ge et al. "Deep Feature Clustering for Seeking Patterns in Daily Harmonic Variations". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–10. ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.2020.3016408.

[15] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (July 28, 2006), pp. 504–507. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1127647.

[16] Han Hu et al. "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial". In: *IEEE Access* 2 (2014), pp. 652–687. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2332453.

[17] *IEC 61000-4-30:2015+AMD1:2021 CSV | IEC Webstore*.

[18] *IEEE Recommended Practice for Monitoring Electric Power Quality*. IEEE. DOI: 10.1109/IEEESTD.2019.8796486.

[19] Poras Khetarpal et al. "Power Quality Disturbances Detection and Classification Based on Deep Convolution Auto-Encoder Networks". In: *IEEE Access* 11 (2023), pp. 46026–46038. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3274732.

[20] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014). DOI: 10.48550/ARXIV.1412.6980.

[21] Hui Liu et al. "Complex power quality disturbances classification via curvelet transform and deep learning". In: *Electric Power Systems Research* 163 (Oct. 1, 2018), pp. 1–9. ISSN: 0378-7796. DOI: 10.1016/j.epsr.2018.05.018.

[22] Mingping Liu et al. "Classification of Power Quality Disturbance Using Segmented and Modified S-Transform and DCNN-MSVM Hybrid Model". In: *IEEE Access* 11 (2023), pp. 890–899. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3233767.

[23] Jian Ma et al. "Classification of Power Quality Disturbances via Deep Learning". In: *IETE Technical Review* 34.4 (July 4, 2017), pp. 408–415. ISSN: 0256-4602, 0974-5971. DOI: 10.1080/02564602.2016.1196620.

[24] Claudia Malzer and Marcus Baum. "A Hybrid Approach To Hierarchical Density-based Cluster Selection". In: *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Sept. 14, 2020, pp. 223–228. DOI: 10.1109/MFI49285.2020.9235263. arXiv: 1911.02282[cs].

[25] Leland McInnes, John Healy, and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: (2018). DOI: 10.48550/ARXIV.1802.03426.

[26] James R. Munkres. *Elements of Algebraic Topology*. 1st ed. CRC Press, Mar. 5, 2018. ISBN: 9780429493911. DOI: 10.1201/9780429493911.

[27] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: (2015). DOI: 10.48550/ARXIV.1511.08458.

[28] Callum O'Donovan, Cinzia Giannetti, and Grazia Todeschini. "A Novel Deep Learning Power Quality Disturbance Classification Method using Autoencoders:" in: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*. 13th International Conference on Agents and Artificial Intelligence. Online Streaming, — Select a Country —: SCITEPRESS - Science and Technology Publications, 2021, pp. 373–380. ISBN: 9789897584848. DOI: 10.5220/0010347103730380.

[29] Roger Alves De Oliveira and Math H.J. Bollen. "Deep learning for power quality". In: *Electric Power Systems Research* 214 (Jan. 2023), p. 108887. ISSN: 03787796. DOI: 10.1016/j.epsr.2022.108887.

[30] Vassil Panayotov et al. "Librispeech: An ASR corpus based on public domain audio books". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ICASSP 2015 - 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). South Brisbane, Queensland, Australia: IEEE, Apr. 2015, pp. 5206–5210. ISBN: 9781467369978. DOI: 10.1109/ICASSP.2015.7178964.

[31] Juan Jose Gonzalez de la Rosa. *Real-life Power Quality Transients*. May 11, 2017.

[32] Anjan Kumar Sahoo and Sudhansu Kumar Samal. "Online fault detection and classification of 3-phase long transmission line using machine learning model". In: *Multiscale and Multidisciplinary Modeling, Experiments and Design* 6.1 (Mar. 1, 2023), pp. 135–146. ISSN: 2520-8179. DOI: 10.1007/s41939-022-00132-x.

[33] Rafael S. Salles et al. "Analytics of Waveform Distortion Variations in Railway Pantograph Measurements by Deep Learning". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–11. ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.2022.3197801.

[34] C.E. Shannon. “Communication in the Presence of Noise”. In: *Proceedings of the IRE* 37.1 (Jan. 1949), pp. 10–21. ISSN: 0096-8390. DOI: 10.1109/JRPROC.1949.232969.

[35] Yue Shen et al. “Power Quality Disturbance Monitoring and Classification Based on Improved PCA and Convolution Neural Network for Wind-Grid Distribution Systems”. In: *Energies* 12.7 (Apr. 3, 2019), p. 1280. ISSN: 1996-1073. DOI: 10.3390/en12071280.

[36] Surendra Singh et al. “Power Quality Detection and Categorization Algorithm Actuated by Multiple Signal Processing Techniques and Rule-Based Decision Tree”. In: *Sustainability* 15.5 (Jan. 2023), p. 4317. ISSN: 2071-1050. DOI: 10.3390/su15054317.

[37] Nagendra Kumar Swarnkar, Om Prakash Mahela, and Mahendra Lalwani. “Multivariable signal processing algorithm for identification of power quality disturbances”. In: *Electric Power Systems Research* 221 (Aug. 1, 2023), p. 109480. ISSN: 0378-7796. DOI: 10.1016/j.epsr.2023.109480.

[38] Bendong Tan et al. “Representational learning approach for power system transient stability assessment based on convolutional neural network”. In: *The Journal of Engineering* 2017.13 (Jan. 2017), pp. 1847–1850. ISSN: 2051-3305, 2051-3305. DOI: 10.1049/joe.2017.0651.

[39] Roman Targosz and Jonathan Manson. “Pan-European power quality survey”. In: *2007 9th International Conference on Electrical Power Quality and Utilisation*. 2007 9th International Conference on Electrical Power Quality and Utilisation. ISSN: 2150-6655. Oct. 2007, pp. 1–6. DOI: 10.1109/EPQU.2007.4424203.

[40] Radovan Turovic et al. “Training an LSTM Voltage Sags Classificator on a Synthetic Dataset”. In: *2021 21st International Symposium on Power Electronics (Ee)*. 2021 21st International Symposium on Power Electronics (Ee). Novi Sad, Serbia: IEEE, Oct. 27, 2021, pp. 1–6. ISBN: 9781665401876. DOI: 10.1109/Ee53374.2021.9628265.

[41] Venkataramana Veeramsetty et al. “Power quality disturbances classification using autoencoder and radial basis function neural network”. In: *International Journal of Emerging Electric Power Systems* (Sept. 25, 2023). ISSN: 1553-779X. DOI: 10.1515/ijeeps-2023-0143.

[42] Shouxiang Wang and Haiwen Chen. “A novel deep learning method for the classification of power quality disturbances using deep convolutional neural network”. In: *Applied Energy* 235 (Feb. 1, 2019), pp. 1126–1140. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2018.09.160.

[43] ZHAO Feng et al. “Classification of multiple power quality disturbances based on CNN-BiLSTM-Attention”. In: 27.7 (July 1, 2024). DOI: 10.6180/jase.202407_27(7).0007.

[44] Lipeng Zhu et al. “Robust Representation Learning for Power System Short-Term Voltage Stability Assessment Under Diverse Data Loss Conditions”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023), pp. 1–13. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2023.3325542.

A Appendix

A.1 Proof of pseudometric

Proof. We need to show that if ℓ is a pseudometric, then

$$d(x_1, x_2) = \min\{\ell(x_1, x_2), \ell(x_1, \sigma x_2), \ell(x_1, \sigma^2 x_2)\}$$

is also a pseudometric.

The operator σ performs cyclic permutations. There are three possible permutations, so σ^3 is the identity operator. Furthermore, $\ell(x_1, x_2) = \ell(\sigma x_1, \sigma x_2)$ since x_1 and x_2 are permuted in the same way.

We need to show the three properties defined in definition 8.

Distance to self:

$$d(x_1, x_1) = \min\{\ell(x_1, x_1), \ell(x_1, \sigma x_1), \ell(x_1, \sigma^2 x_1)\} = \ell(x_1, x_1) = 0.$$

Symmetry:

$$\begin{aligned}
d(x_1, x_2) &= \min\{\ell(x_1, x_2), \ell(x_1, \sigma x_2), \ell(x_1, \sigma^2 x_2)\} \\
&= \min\{\ell(x_2, x_1), \ell(\sigma x_2, x_1), \ell(\sigma^2 x_2, x_1)\} \\
&= \min\{\ell(x_2, x_1), \ell(\sigma^3 x_2, \sigma^2 x_1), \ell(\sigma^3 x_2, \sigma x_1)\} \\
&= \min\{\ell(x_2, x_1), \ell(x_2, \sigma^2 x_1), \ell(x_2, \sigma x_1)\} \\
&= \min\{\ell(x_2, x_1), \ell(x_2, \sigma x_1), \ell(x_2, \sigma^2 x_1)\} \\
&= d(x_2, x_1)
\end{aligned}$$

Triangle inequality:

$$\begin{aligned}
d(x_1, x_3) &= \min\{\ell(x_1, x_3), \ell(x_1, \sigma x_3), \ell(x_1, \sigma^2 x_3)\} \\
&\leq \min\{\ell(x_1, x_2) + \ell(x_2, x_3), \ell(x_1, x_2) + \ell(x_2, \sigma x_3), \ell(x_1, x_2) + \ell(x_2, \sigma^2 x_3)\} \\
&= \ell(x_1, x_2) + \min\{\ell(x_2, x_3), \ell(x_2, \sigma x_3), \ell(x_2, \sigma^2 x_3)\} \\
&= \ell(x_1, x_2) + d(x_2, x_3) \\
&\leq d(x_1, x_2) + d(x_2, x_3)
\end{aligned}$$

□

A.2 Classification metrics

The classified samples from a binary classifier can be separated into the categories

- True positives (TP): Positive samples that are classified as positive
- False positives (FP): Negative samples that are classified as positive
- True negatives (TN): Negative samples classified as negative
- False negatives (FN): Positive samples that are classified as negative

From these four integers, a couple of metrics can then be derived

- Precision: $P = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$
- True positive rate (recall, sensitivity): $\mathrm{TPR} = R = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$
- False positive rate: $\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP} + \mathrm{TN})$
- Accuracy: $A = (\mathrm{TP} + \mathrm{TN})/(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})$

which all measure different aspects of a model. Accuracy is a common metric when one is equally interested in classifying positive and negative samples, however, one drawback is that it is very easy to achieve high accuracy for unbalanced datasets, as is easiest demonstrated using an example.

Example 6. *In a classification problem with* 99 *negative samples and* 1 *positive sample, a classifier that always outputs a negative answer will achieve 99% accuracy, even though it might in practice be better to have some false positives in order to get more true positives. Medical applications immediately come to mind.*

An alternative metric is the F_1*-score*, which is more commonly used with unbalanced datasets where positive samples are rare, defined as

$$F_1 = \frac{2TP}{2TP + FP + FN}. \tag{A.1}$$

In the example above, the F_1-score would become 0, since there are no true positives. For multi-class classification, each class counts as the positive class in relation to all the others. A simple generalisation of the F_1-score is to take the average of the F_1-score for each class, called the *macro* F_1*-score*:

$$F_M = \frac{F_1(1) + ... + F_1(C)}{C} \tag{A.2}$$

where $F_1(i)$ is the F_1-score for class i. The macro F_1-score for the example above would become approximately 0.25.

For a classifier that outputs probabilities instead of a binary output, the numbers TP, FP, TN and FN will depend on what threshold is chosen for determining when a sample is classified as positive or not. Intuitively, a low threshold will result in a high number of false positives, leading to low precision but also high recall. In contrast, a high threshold will cause the model to output very few positive samples, resulting in high precision but low recall. By changing the threshold, we can obtain metrics such as precision and recall as a function of the threshold. These can then be plotted in a precision-recall plot or in a receiver operating characteristic (ROC) curve. A ROC curve plots the true positive rate against the false positive rate and typically looks like Figure A.1. As the threshold is decreased, the true positive rate increases, at the cost of a higher false positive rate as well. One advantage of the ROC curve is that a scalar metric can be extracted directly from it, namely the *area under*

curve (AUROC). Good classifiers are able to achieve high true positive rates for each given false positive rate, increasing the AUROC up to 1, compared to random classifiers that have AUC 0.5. AUROC also has a probabilistic interpretation: it is the probability that the classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample.

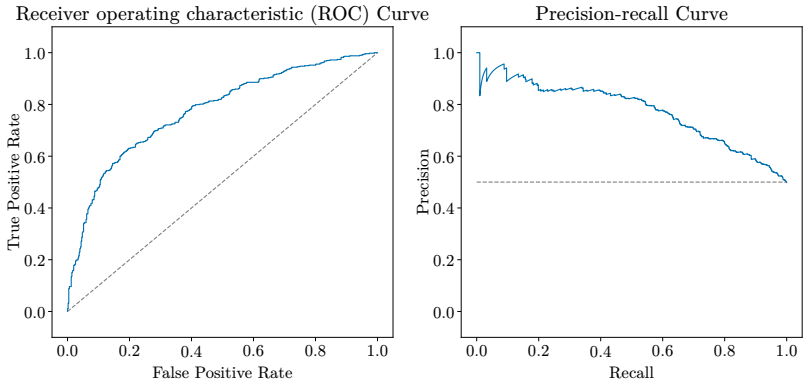

Figure A.1: A ROC curve and a precision-recall curve.

CHALMERS
UNIVERSITY OF TECHNOLOGY