Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Master's Thesis

# Automatic feature extraction for time series classification

*Janhavi Puranik*

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr Paul G. Ploger
Prof. Dr. Sebastian Houben
Dr. Anastassia KÂĺustenmacher

March 2022

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

| | |
|---|---|
| Date | Janhavi Puranik |

# Abstract

Your abstract

# Acknowledgements

I would like to thank my supervisors Prof. Dr. Paul G. Ploger Prof. Dr. Sebastian Houben and Dr. Anastassia KÃijstenmacher for providing the opportunity to work on this Research and Development Project. I would like to especially thank Prof. Dr. Sebastian Houben and Dr. Anastassia KÃijstenmacher for supporting and motivating me throughout the project.

I would like to thank Satiya Ramesh, Proneet Sharma, and Deepan for their immense support while conducting the experiments. I would like to also thank Satiya Ramesh for her valuable suggestions to improve this report. Finally, I extend my gratitude to my family and friends for their enduring support, undying inspiration, and endless encouragement.

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

In the automobile industry Artificial intelligence has become an essential tool for improvement in production and manufacturing processes as well as the built in functionalities of the vehicle. Machine learning can help in providing vehicle maintenance related recommendations which can detect abnormal functions at early stage and ensure safety of the driver. The maintenance model supported by the current vehicle is static scheduled method that is, there is a fixed time at which a vehicle is checked for problems. Instead a predictive model based on the data points collected from manufacturing, service providers and the vehicle running on road can be used. The data collected is usually from sensors that play extensive role in capturing, visualizing and storing the variables of processes mentioned above.

The problems mentioned above can be broken down to detecting the abnormal behavior in the vehicle functionalities using the data collected from sensors. To detect the abnormal behavior data points that do not belong to defined normal behavior have to be selected. This becomes difficult as normal behavior is evolving in nature. According to survey by [13] there are many challenges that are faced when detecting anomalies in industrial data. Firstly, the boundary between normal and anomalous behavior is continuous. Hence many a times the assignment of the data points that lie near the boundary can be falsely assigned. Second, the data collected is from the sensors which implies that the data will contain noise. Often it is difficult to differentiate the outliers from the noise in such situations. Another issue is anomaly detection needs a problem specific definition which cannot be generalized. Biggest challenge being that often the data is not labeled as it can be unavailable or expensive to acquire. Thus to solve the anomaly detection a problem specific solution based on the nature of data and availability of labels is used. Different techniques such as statistical methods, machine learning, spectral theory can be applied based on the problem definition to detect outliers.

To monitor the process data is collected at specific time intervals from the sensors. This implies that the data is in form of time series. To detect the anomalous behavior points or patterns that do not follow the normal area have to be identified. There is no exact notion on to what can be called as anomaly because the sensor behavior cannot be fixed. Thus all specific applications that are based on machine learning are built. As mentioned above the labels are not available hence mostly unsupervised methods are applied. The time series in its crude form from the sensors contains redundant information and noise which might not be relevant to train a model. Therefore the data is subjected to preprocessing such as filtering and feature extraction.

Recent studies of fault diagnosis have demonstrated that identifying features that convey fault information are crucial for detecting abnormalities in the process. Commonly, feature extraction methods involve identifying a fixed set of features from the data based on domain. These features are normally based on time, frequency, or time-frequency measurements. Signal amplitude values that calculate time-domain features. These features are easy to obtain and assume the signal to be stationary. This is one of the disadvantages of the method. Time series are often converted to the frequency domain which enables easy identification of the oscillations in the series and their amplitudes and phases. The disadvantage of this method is extracted features can be dependent on each other and the method is computationally heavy. Therefore a automatic way of identifying these features is required which can be done with help of deep-learning methods.

Through this work, we are trying to explore different approaches and libraries present to extract the relevant features from the sensor automatically to identify the abnormalities in data efficiently.

## 1.1 Motivation

To assure the safety of the passenger, one of the major elements which need special focus is the proper closing of is door. In many researches the analysis of this problem is done using the sound of the door closing. But in order to correctly measure sound without any noise interfering with the data collection is a tedious task. To avoid this problem accelerometers can be used to calculate the maximum velocity at which the door closes properly. Analysis of this data is required in during manufacturing to assure the quality of produced product and also for daily usage of car as there can be some wear and tare over time. This process not only assures the safety of the passenger but acts as a monitor to usage and saves maintenance cost. In manufacturing it acts as monitoring agent to assure quality of production and repeatability.

## 1.2 Challenges and Difficulties

This section mentions some of the challenges in feature extraction of time series data. It is difficult to select an appropriate set of features for given data set. Following problems are faced when extracting relevant set [14]:

- Unifying method missing: which methods are yet to be explored out of methods present to create a relevant set of features
- Variety of feature sets: classification task demands a different set of features which is not fixed for every method
- Large feature space: There is no standard feature set from which a subset is selected for the further task

## 1.3 Problem Statement

Feature extraction is done to enhance the representation of the dataset by additional descriptive variables. These variables are expected to increase the accuracy of the predictive models. Autoregressive

models and transformation methods like Fast Fourier Transform (FFT) [15] or Discrete Wavelet Transform (DWT) [16] are the most commonly used methods for feature extraction of time series data. Wherein the physical attributes are extracted as additional attributes of data. These methods use handcrafted features and need manual computation. To automate this process, libraries like tsFresh [17] have been built. The limitation of these methods is that the complexity increases with the dimension of the data and also many redundant features are created. Complex time series data which is used in this project is not all that expressive. Detecting faults using one particular feature is not possible. From previous analysis of the data, manual selection of physical features is required to classify the signals. Thus the first method is applied to extract features like statistical, temporal, FFT from the raw signals and it needs manual calculations and interpretation. Further experimentation to identify the number of features to be selected according to the importance and data balancing will be done. One of the issues with this method is that creating important features with great predictive power for the problem are ignored.

Deep-learning methods such as Auto-encoders, Convolution neural network (CNNs), Echo state network (ESN) and Long Short-Term Memory (LSTM) have been used to extract features from sensor signals [18]. The problem while using such deep-learning methods directly for feature extraction of comfort door dataset is that, it is complex and does not have many samples to train the model. Therefore in such a case transfer learning method can be employed which enables the reusability of the existing models, along with its parameters. First suitable models for transfer learning will be selected. Selected models will be modified and experiments will be conducted to look for suitable architecture for training comfort door dataset. Both methods will be validated on open-source data. Comparative evaluation of the two models implemented will be done to identify which model is suitable for feature extraction of the comfort door dataset based on increasing the classification task accuracy.

# 2

# State of the Art

Methods for detecting a fault in signals can be classified into two types model-based approach and feature-based approach. In the model-based methods, a time-ordered stochastic process is used to describe the observations. This method is sensitive to noise and the fault model that describes the characteristic fault in signal must be defined. This information is generally not available in real-time data thus in such cases feature-based approaches are preferred. In this approach features are considered as random set variables and extracting features and selecting feature subsets are crucial steps. Which in turn reduces the number of attributes or data dimensions considered when making a decision. This work focuses on feature extraction of time series data and in the following sections detailed literature review is presented.

## 2.1 Feature extraction

The main goal of feature extraction is to reduce the dimensionality of data and retain important information of the original data. This method makes use of characteristics that are effective visual indicators of distinct populations and the physics that underpin them. The process of feature extraction reduces the redundant information and eliminates noise in-turn reducing the time for time-series classification. This approach is specifically built for discriminative models of classification where the relation between the time series features and probability distribution of classes is learned [2]. There are various techniques available which are discussed in following sections.

### 2.1.1 Domain specific methods

To analyze time series data both time domain and frequency domain methods are available. These features are handcrafted with help of the domain knowledge and then used as discriminators in classification models. Multiple features are required to model the data for classification, if only one feature is used it can lead to high uncertainty and misclassification of data. When the length of the signal is small, the time domain features suffice the analysis but as the length increases to reduce the dimensionality frequency domain features are suitable [19]. One of the early methods temporal features such as min, max and skewness were extracted for data as in [20]. In [21] the maximum and minimum peaks of the time series were extracted to observe the pattern in audio data. A fixed set of domain-specific features have been extracted for example time-domain features such as mean, variance etc. Statistical features are calculated

from the raw data to create the compressed representation. Several papers suggest that these features can be used to recognize patterns in time series. Statistical features such as skewness, kurtosis and entropy can be used to classify audio signals in [22]. Feature extraction can be applied to whole data or by dividing it into windows and creating representation for each. There have been several studies that based the classification on the mean, standard deviation, skewness, and kurtosis of the time series [20]. A similar approach is defined in the Highly comparative feature-based method by [23]. -Different systems implement the highly comparative approach. Tsfresh [17] is currently one of the most popular tools. The appropriate features are selected using the FRESH algorithm (Feature Extraction using Scalable Hypothesis tests). It performs the following tasks: '

- Testing hypotheses to determine the level of dependency between each label and the target labels.
- Features are selected based on p-values computed for each of them.

Along with this approach to fuse data from multiple sensors and creating a feature vector a library named Tsfuse is present [24]. Commonly used feature extraction in frequency domain is using Fast Fourier transform details of which have been described in chapter 3 and the application can be seen in paper [25]. Another domain specific method which provides information on both domains is wavlet transform it eliminates drawbacks of conventional Fourier transform. It has been successfully implemented in [26] which provides promising results.

Main issue with the domain specific methods is generalization of the problem is difficult for example the features that work for ECG data might not be effective for sound signals.The process is time consuming and expensive to calculate. Another issue related to this kind of method is it requires labeled data sets which is sometimes not available in real time data.

## 2.2 Deep-learning

To overcome the short comings of the domain specific methods unsupervised learning algorithms can be applied for feature extraction [27]. Also the deep learning methods are comparatively better at modeling complex real time data. Deep networks are designed to separate the factors that affect input data variation at the lower layers and combine the representations at the higher layers. There are various deep-learning models that have been investigated for feature extraction of time series data most commonly used is Autoencoder as it can be seen in the paper [28]. Many implementations such as [29], [30] are available for modeling the data using LSTMs and Recurrent neural network(RNN) to model time-series data as these models are suitable for sequential data. The problem with RNNs is that they have difficulty in learning longterm defficencies due to the vanishing gradient problem [31] which can be solved by LSTMs. According to [32] LSTMs are not that effective to model this type of data. [33] examined implementation of Deep Belief Network (DBN) on sensor data by Human activity recognition data [34]. The method presented here does not have a effective signal processing unit and the labels of data are ignored. To overcome the ignorance of patterns which was seen in DBN deep neural network architectures which is popularly used for image classification data can be used that is CNN [35]. This implementation states that CNN can serve as a competitive tool of feature learning and classification in comparison to the sate

of art feature engineering techniques. Overview of methods that can be used for feature engineering of signal data is shown in figure 2.1.

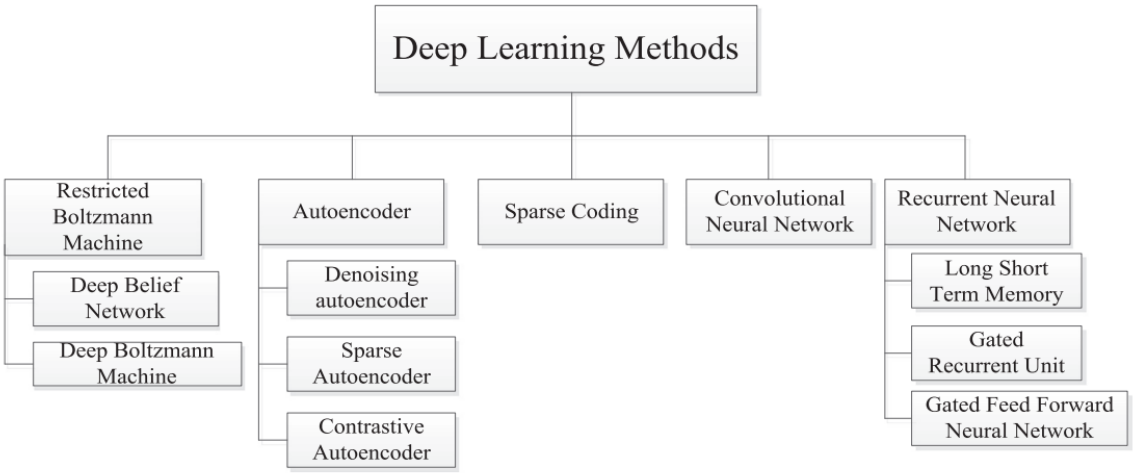



Figure 2.1: An overview of the different deep learning approaches for time series classification [1]

## 2.3 Classification

The summary of classification algorithms is depicted in figure 2.2. This thesis focuses on the Discriminative model of classification.

### 2.3.1 Domain specific

### 2.3.2 Image transforms

Instead of direct use of raw signal as input to the deep learning classifiers the signals can be encoded into two dimensional format. Many different approaches have been used to analyse and convert the signal in image format one of the poular is recurrence plots that help to analyse the next time step as can be seen in the implementation of [36]. Other types of image representations that have been reaserched are Gramian fields and Markov transition fields [37].

Another method of image transformation that uses both time and frequency domain information to represent the signal data is spectrogram which has been popularly used as input to classification for audio data. The implementation has also been effectively implemented for the human activity recognition data and ECG medical datasets.

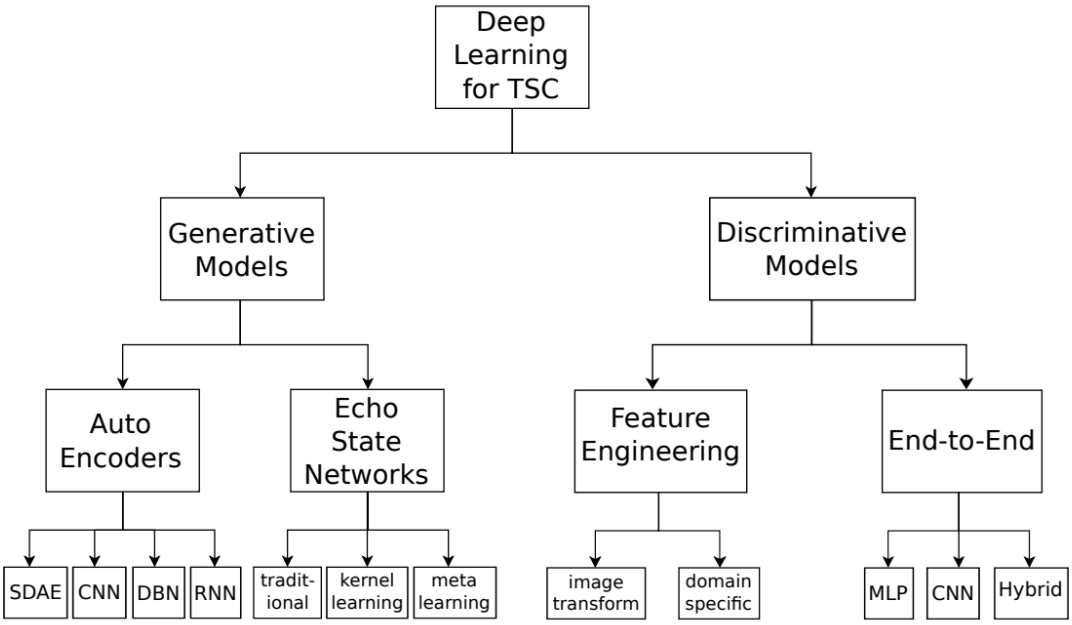



Figure 2.2: An overview of the different deep learning approaches for time series classification [2]

### 2.3.3 Transfer learning

### 2.3.4 Anomaly detection

# 3

# Background Knowledge

## 3.1 Feature extraction

### 3.1.1 Domain-Specific Features

Standard machine learning classification algorithms cannot be directly applied to raw time series. Hence, features have to be extracted. As the concept of input and output features is missing in time series, variables should be chosen from it in such a manner that they aid to better classification of data. By evaluating the temporal relation between data points, time-domain processing better understands those relationships. If the information of interest repeats at regular or semi-regular intervals, a simple transformation can convert the time domain information to the frequency domain. This conversion isolates vibrational information for analysis within and between vibrational frequencies present in the time series. Following the domain, specific features can be extracted from the time-series data:

#### Temporal

Temporal features define the ratios of relative intervals between events. The information about the frequency and sequence is not present in these features. These features are easy to extract and have simple physical interpretation, like: the maximum amplitude, minimum energy, centroid, mean absolute etc [38].

#### Statistical

These features are obtained by computing the statistic on the values of time-series. For example Kurtosis, Skewness, Maximum, Minimum, Mean, Median, etc.

#### Spectral

These features can be obtained by converting the timeseries to frequency domain by using methods like Fourier transform. They help in analyzing patterns in the signal. For example fundamental frequency, frequency components, spectral centroid, spectral flux, spectral density, etc [38].

### 3.1.2 Time domain transformations

Transforms in the time domain are similar to magnitude domain transformations, except they occur on the time axis. This means that the elements are displaced to different time step. There are various transformation described as follows

#### Slicing

Slicing of time-series is similar to cropping a image, that is either removing the end elements from the signal or selecting a particular pattern. This is represented in the figure 3.1. This method can be used to select required information from the raw data or each slice from time series can be used in classification.

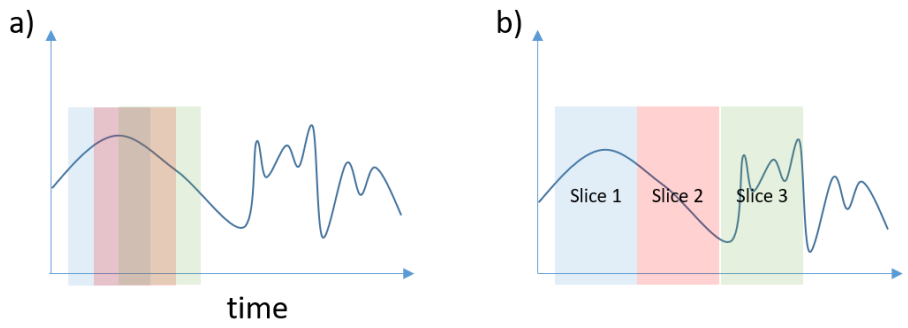



Figure 3.1: Window slicing [3]

#### Fourier transform

Fourier transform is a tool that converts the time-series into alternate representation, characterized by the sine and cosine functions of varying frequencies. According to this theory any complex function can be represented as the combination of periodic functions which is known as Fourier series.

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n cos(\frac{2\pi n}{T}x) + b_n sin(\frac{2\pi n}{T}x) \tag{3.1}$$

**Missing annotations** Any periodic time-series can be represented as sum of sinusoidal components which is given by the equation 3.1. Time series is transformed into Fourier series by measuring every possible cycle and returning the amplitude, offset and rotational speed for every detected cycle. One of commonly used methods to achieve the transformations for real world signals is Fast Fourier transform (FFT). It is a optimized algorithm for obtaining Discrete fourier transform. A signal's frequency components are derived from sampled data over a period of time. Each component of a signal is a single sinusoidal oscillation with its own amplitude and phase. Figure 3.2 shows a example of transformation.

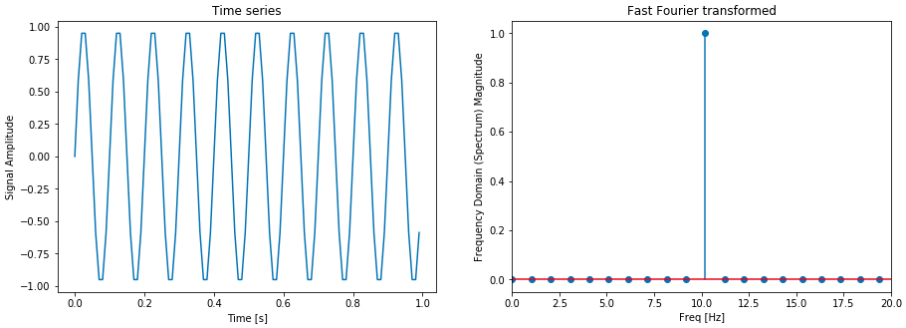



Figure 3.2: Fast Fourier transform example [4]

### Spectrogram Representation

Spectrogram is a visualization method where the temporal and spectral components of a signal can be viewed simultaneously. A spectrogram can be constructed using the short-time Fourier transform, that is a time window is determined, and then translated across the field. By calculating the Fourier transform of each windowed field, the spectrogram is created. As it provides a analysis of time-frequency simultaneously it can be used to identify the nonlinearities in signal. Thus it proves a important tool for analysis of real world data such as sound. Figure 3.3 shows a example of spectrogram.

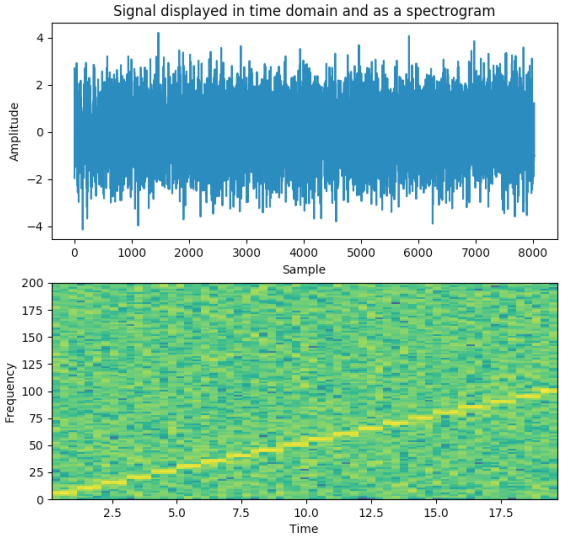



Figure 3.3: Spectrogram plot example [5]

Consider we have a signal x of length N. If we create consecutive segments of x having length of m where $m << N$ and $X \in R^{m \times (N-m+1)}$ are the consecutive columns of the matrix. That is, $[x[0], x[1], ..., x[m-1]]^T$ is the first column $[x[1], x[2], ..., x[m]]^T$ is second and so on. The rows and colums of this matrix are indexed by time. As we can observe that matrix X is repetitive representation of x. So

the spectrogram of signal is calculated by applying DFT to the columns of $X$ and obtaining a matrix $\hat{X}$. Therefore,

$$\hat{X} = \bar{F}X \tag{3.2}$$

$$X = \frac{1}{m}F\hat{X} \tag{3.3}$$

In the matrix $\hat{X}$ columns are indexed by time and rows by frequency. So each value is representation of a point in frequency and time. $\hat{X}$ is invertible due to the transformations performed above. The information in this matrix is also highly redundant. The matrix is spectrogram and can be visualized with help of a plot.

### 3.1.3 Deep-learning based methods

**Auto-encoder**

Autoencoders are type neural networks which learn data encodings in unsupervised way. The network consists of two parts encoder and decoder. This encoder-decoder system learns efficiently through optimization process, that is when the data is fed to the network its output is compared to initial data and the error is backpropagated through the architecture to update the weights at each iteration. Three components of the autoencoder as seen in figure 3.4 are described as follows:

1. **Encoder**: In this module the input data is converted to encoded representation by reducing the dimensionality of data. In terms of neural networks the encoding is accomplished by reducing nodes in subsequent layers. The process reduces the information and only the most relevant or average information is forwarded.

2. **Latent space**: This space contains the reduced or compressed knowledge of the data. The representation obtained can be used as feature input to classification algorithms.

3. **Decoder**: This module reconstructs the encoded information of the data to it original, that is it decompresses the data.

As discussed above the objective of autoencoder is to have input to be equivalent to output. The encoder function $\phi$ maps the data to latent space $F$ and decoder function $\varphi$ maps it back to original data $\chi$ as shown in following equations:

$$\phi : \chi \to F \tag{3.4}$$

$$\varphi : F \to \chi \tag{3.5}$$

$$\phi, \varphi = argmin \left\| \chi - (\phi \cdot \varphi)\chi \right\|^2 \tag{3.6}$$

The encoder network is modeled by passing the standard neural network function into activation

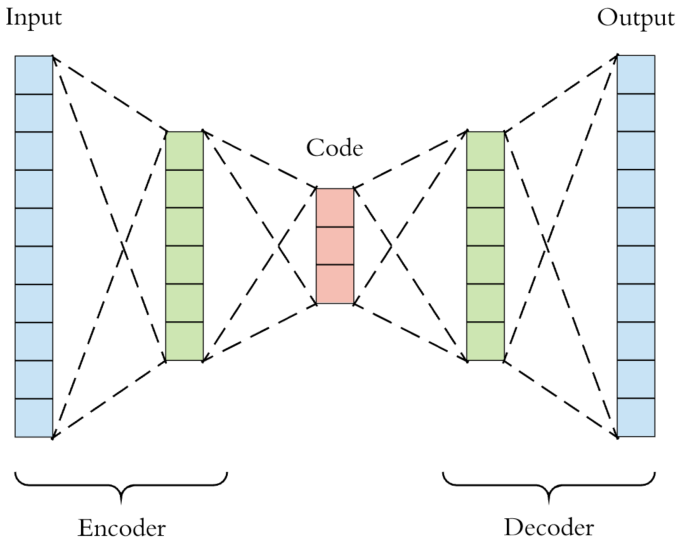



Figure 3.4: Autoencoder [6]

function given in by equation 3.7.$z$ here represents latent dimension

$$z = \sigma(Wx + b) \tag{3.7}$$

The decoder function can be represented similarly with help of different set of weights and biases and activation function.

$$x' = \sigma'(W'z + b') \tag{3.8}$$

The loss function can be written as:

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \mathbf{MSE}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \|\mathbf{x} - \hat{\sigma}\{\hat{\mathbf{W}}\,[\sigma(\mathbf{W}\mathbf{x} + \mathbf{B})] + \hat{\mathbf{B}}\}\|^2 \tag{3.9}$$

There are various implementation available for feature extraction using this method to encode the features of data instead using hand crafted method [39]. The deep autoencoders are designed in a fashion to reduce the dimensionality of raw sensor data [40] and transform them into feature vector with consized information on data. Variation of implementations can be found in the literature with a aim to find robust representation of data.

### Long Short Term Memory networks(LSTM)

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks (like CNN), LSTM has feedback connections. Because of the feedback connections, LSTM can not only process single data points (like image) but also a sequence of data points like a sentence or a time series.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell

remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. In the image below, the $C_t$ is the cell state and it runs down the information through the entire chain. On this, the information is removed or added by Forget gate or Input gate.

1. The Forget gate: The first step in LSTM is to decide what information to remove from the cell state. This decision is made by a sigmoid layer called the forget gate layer. It looks at $ht-1$ and xt and outputs a number between 0 and 1 for each number in the cell state $C_{t-1}$ . A 1 represents completely keep this while a 0 represents completely get rid of this. The output of the forget gate is calculated using the following equation:

$$f_t = \sigma(W_f.[h_{t-1}, x_{t-1}] + b_f) \tag{3.10}$$

2. Input gate: The next step is to decide what new information we are going to store in the cell state. For this step, first the sigmoid layer decides which values to update. Then a *tanh* layer creates a vector of new values that are to be added in the cell state. Then these two are combined to create an update to the state. Mathematically it represented by:

$$i_t = \sigma(W_i.[h_{t-1}, x_{t-1}] + b_i) \tag{3.11}$$
$$C_t = tanh(Wc.[h_{t-1}, x_{t-1}] + b_c) \tag{3.12}$$

   The updated cell state is calculated by: $C_t = f_t * C_{t-1} + i_t * \bar{C}_t$

3. Output gate: In the next step, the output is calculated. This will be a filtered version of the cell state. First, a sigmoid layer is used which decides what parts of the cell state are going to output. Then, the cell state is put through *tanh* (to push the values to be between $-1$ and 1 ) and multiply it by the output of the sigmoid gate, so that only the decided parts are output. Mathematically it is represented as:

$$o_t = \sigma(W_o[h_{t-1}, x_{t-1}] + b_o) \tag{3.13}$$
$$h_t = o_t * tanh(C_t) \tag{3.14}$$

In the above equations, the lowercase variables represent vectors. Matrices Wq contains the weights where the subscript q can either be the input gate i, output gate o, the forget gate f or the memory cell c, depending on the activation being calculated. A vector notation was used which means ctRh is not just one unit of one LSTM cell, but contains $h$ LSTM cell's units.

**CNN**

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which is most commonly applied to analyse visual imagery. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organisation of the Visual Cortex. In

**Variables** [ edit ]

- $\overrightarrow{x_t} \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in (0,1)^h$: forget gate's activation vector
- $i_t \in (0,1)^h$: input/update gate's activation vector
- $o_t \in (0,1)^h$: output gate's activation vector
- $h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit
- $\tilde{c}_t \in (-1,1)^h$: cell input activation vector
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training

where the superscripts $d$ and $h$ refer to the number of input features and number of hidden units, respectively.

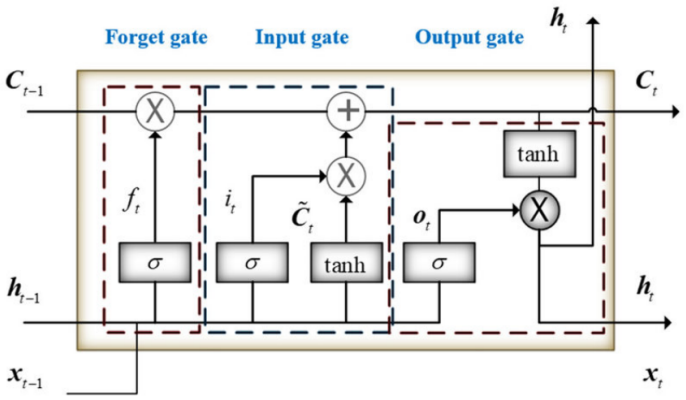



Figure 3.5: The repeating module in an LSTM

humans, Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

As compared to multilayer perceptron (fully connected networks in which each neuron in one layer is connected to all neurons in the next layer), CNN takes advantage of the hierarchical pattern in data. They assemble the patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimise the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered.

A CNN consists of the input layer, the hidden layer and the output layer. The hidden layer of a CNN consists of convolution layers, pooling layers and fully connected layers. Input layer: the input layer of a CNN is a tensor of the image with a shape: (number of inputs) x (input height) x (input width) x (input channels). Hidden layer:

1. Convolution layer: The element involved in carrying out the convolution operation in the Convolutional Layer is called the Kernel/Filter. The filter slides over the input matrix with a certain Stride Value and at each step a dot product is performed. Filter moves to the right till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed. This process generates a feature map which is the input to the next layer. In a CNN, multiple convolution layers are present and at each convolution layer, the feature map complexity increases.

2. Pooling layer: Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

The Convolutional Layer and the Pooling Layer together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-level details even further, but at the cost of more computational power. After going through these layers, the input image is converted into a suitable form for Multi-Level Perceptron (Fully connected layer). The image is flattened into a column vector and is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

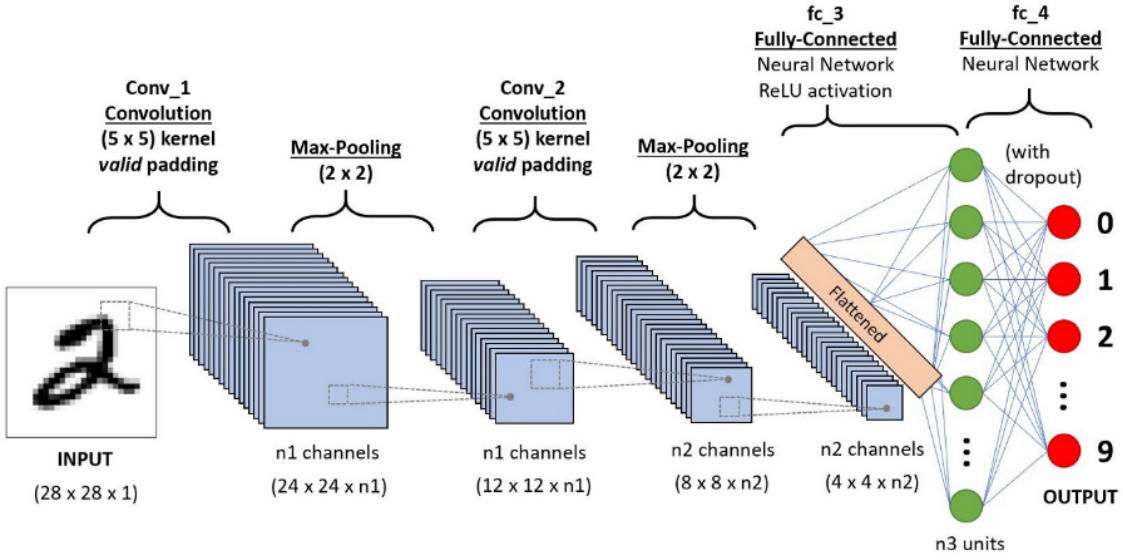



Figure 3.6: CNN architecture for image classification [7]

### 3.1.4 Transfer-learning

A transfer learning method is a machine learning approach that reuses a model developed for one task as a starting point for a second model. Usage of pre-trained models is widely used as they can be used as a start point for solving computer vision or natural language processing tasks. The reason is that developing neural network models for these tasks requires significant amount of computation and time resources. There are two ways to apply transfer learning models described as follows:

**Develop Model Approach**

1. First step in this approach is to select predictive model such that is trained with large amount of data. Also there should be some relation between the data we want to implement this model for and the concept for which the model is trained.
2. After completing the first task, you need to develop an effective model. There must be some feature learning done in the model compared with naive models.
3. Modeling the second task of interest can then be based on the model fit on the source task. The model may be used in its entirety or in parts, depending on the modeling method used.
4. In some cases, the model may need to be customized or refined according to the input-output pairs available for the particular problem at hand.

**Pre-trained Model Approach**

1. From the available models, a pre-trained model can be selected. There are many sources available from research institutions from where a model can be selected according to the problem requirements.
2. After finding a suitable model it can be used as a starting point for second task. Depending on the task a part of model or the entire model can be used.
3. Lastly, the model may need to be customized or refined according to the input-output pairs available for the particular problem at hand.

For this project the transfer-learning is done by using spectrogram as input because the comfort door data is a complex industrial data.

### 3.1.5 Transfer-learning models for images

**VGG**

As it can be visualized in the figure 3.7 this network uses a simple design by stacking 3ÃŮ3 convolutional layers in depths of increasing magnitude. This network uses max pooling to reduce volume size. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier [8]. According to

findings of [8] in 2019 the network was too deep hence hard to train. Thus to make the process of training easier the concept of pre-training is used. In this process step by step training was output of smaller networks was fed to larger as initial step and so on so forth. Even though the pre traing is a logical step it is a tedious task and requires large computation time.

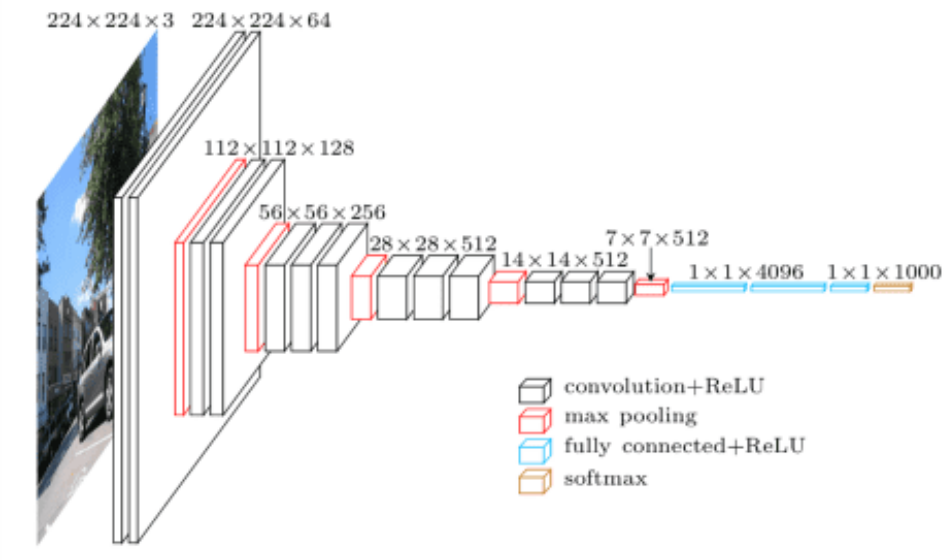



Figure 3.7: VGG network architecture [8]

Disadvantage of using VGG are as follows:

1. It trains very slowly
2. In terms of bandwidth the weights of network are very high.

**Resnet**

With the increase in depth of layers the training becomes a tedious task as seen in section above. Also with the depth there the networks face vanishing gradient problem a problem. As a result the performance gets saturated along the depth. Therefore to solve this problem Resnet provides a identity shortcut connection that is, it skips one or more layers as can be seen in figure 3.8.

The advantage of using the skip layer is if a layer affects the performance of the network it is skipped. The basic idea behind the skip layer is that it is easier to learn the value of $F(x)$ to be zero so it acts like identity function rather than learning the identity function. So this block is also known as the identity block. Another element of the residual network is convolution block. These blocks help in reshaping the output from the first layer to the third layer. Both of these components of Resnet help it in gaining high accuracy and optimization level.

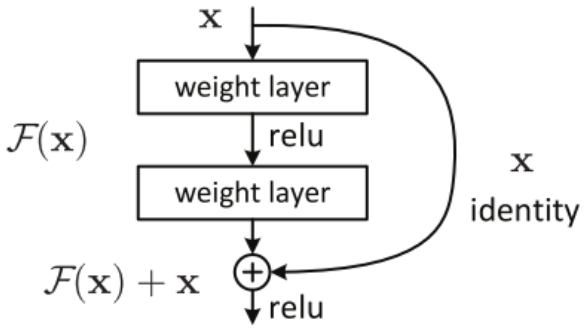



Figure 3.8: Residual learning: a building block [9]

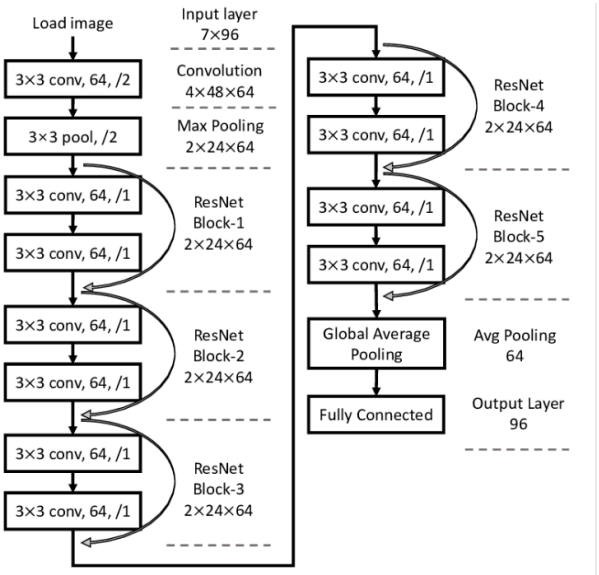



Figure 3.9: Resnet architecture [10]

# 4

# Methodology

This section is written in reference to previous work from a colleague and the information obtained from the meaX company on to the BMW dataset.

## 4.1 Data collection setup

To monitor proper closing of the car door accelerometer is used which is placed on the door with help of a suction mechanism which can be seen in figure 4.1a. The sensor collects acceleration data in three directions, axis of which are marked in the picture as X to be green, Y as blue and Z as red. The data recording is started before the door bumps into the rubber(marked in red in 4.1b) and stopped when door is still.

(a) Car door with sensor

(b) Car door sealing

Figure 4.1: Setup for acceleration measurement

The above experiment is carried out for all the four doors of car and six car models. The data collected is labeled according to car model and the door. The recording are stored in Labview format that is TDMS. The reading is 6000ms long which can be seen in 4.2.

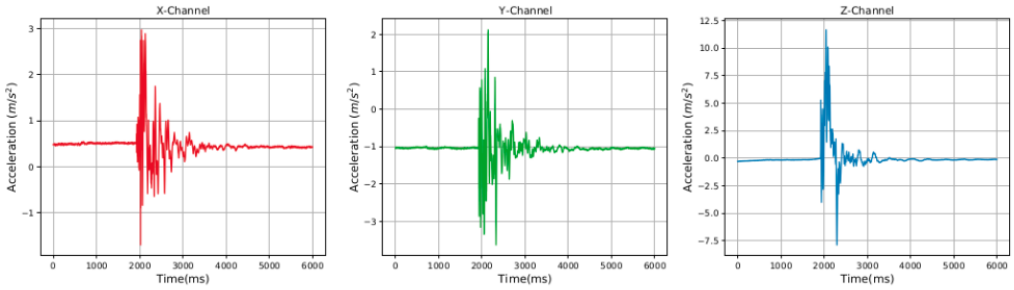



Figure 4.2: Example of X, Y, Z channels of data

## 4.2 Data analysis

The data collected from sensor is distributed sequentially, that it is a time-series data. As mentioned above there are three channels X, Y and Z. For this experiment the sensor measurements in Z direction are considered for following reasons:

- There is no motion in X direction as the door of car is hinged.
- The reading in Y direction account for gravity.
- X and Y channels might have false readings due to sensor attached manually.

The data in Z direction is cleaned and downsized to 3500 samples per signal. To analyze the data further and have additional information on to the door closing the acceleration signal is integrated to obtain the velocity and position. Figure 4.3 shows example of signal samples of acceleration, velocity and position along Z axis.

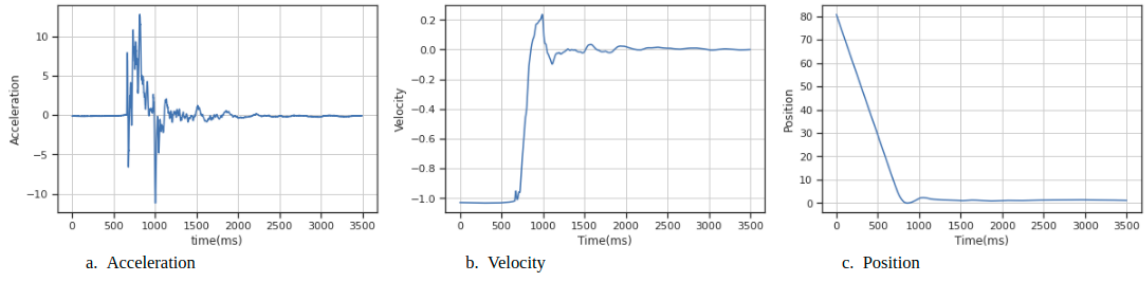



Figure 4.3: Example of acceleration, velocity and position along Z axis

### 4.2.1 Data distribution

The data provided is for six models of BMW cars that can be seen in figure 4.4. As it can be observed from the plot data is highly imbalanced, that is number of negative data samples is very low in comparison to the positive data.

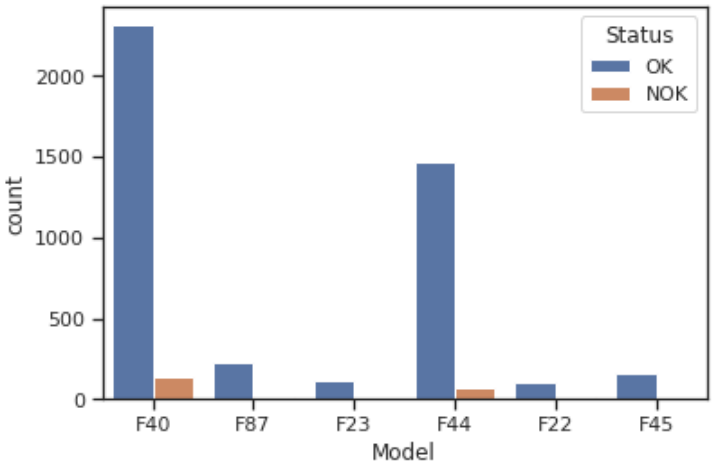
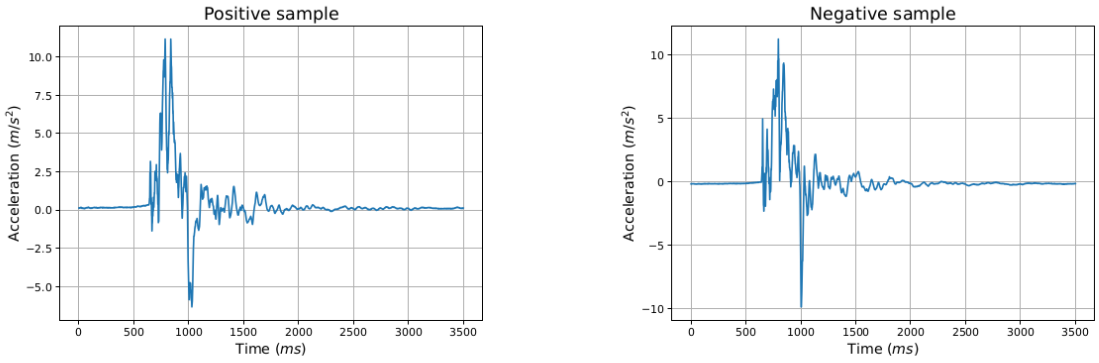



Figure 4.4: Data distribution plot

The signals samples for a positive and negative instance is depicted in figure 6.1. As it can be seen there is no visible difference in the between the instances. This is one of the problems that is addressed in this work.




(a) Positive sample

(b) Negative sample

Figure 4.5: Positive and negative samples

## 4.3 Manually selected features

According to the information provided by the company there are two distinct features maximum velocity and penetration which have been used to classify the data. These features can be visualized in figure 4.6. According the previous work use these features have given better results for classification problem.

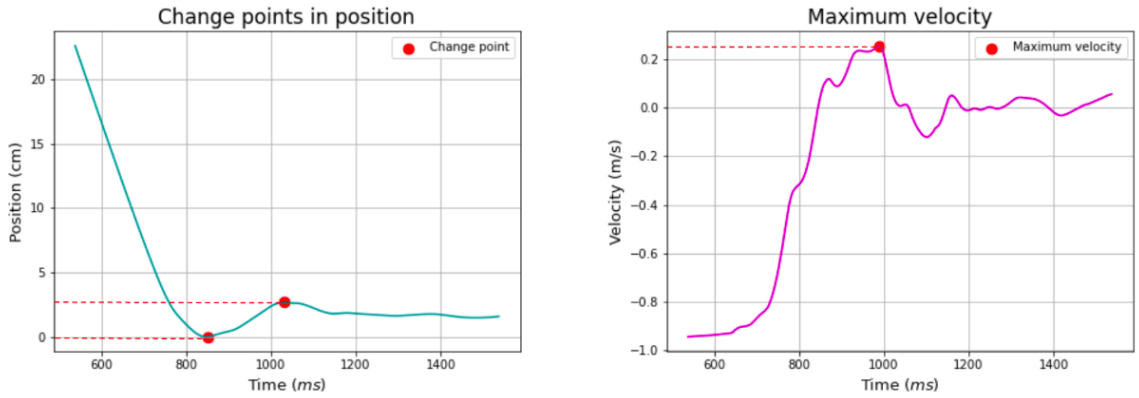



Figure 4.6: Manual features

## 4.4 Data preprocessing

### 4.4.1 Data cleaning

The data is recorded into LabView TDMS format and a document file with additional information of the maximum velocity and penetration is provided. To simplify the usage important information is selected and saved in pickle format, header of the file can be seen in figure 4.7.

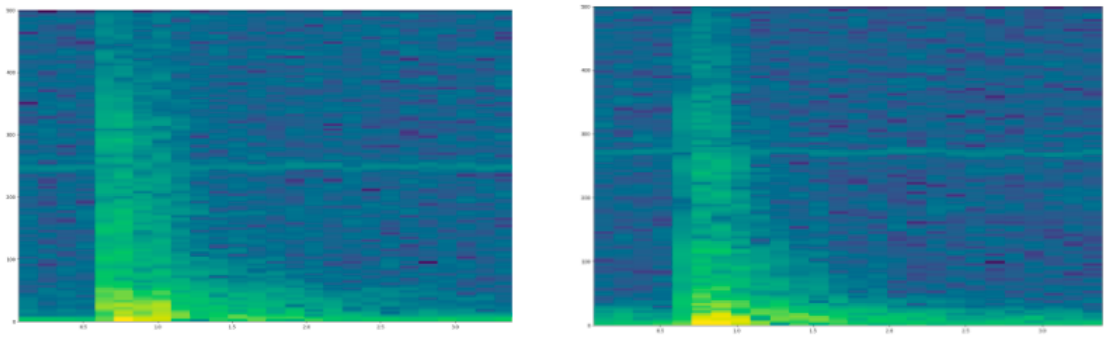
| | Messungid | Model | Door | Status | Closing speed | Penetration | Acc | Vel | Pos |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1179308 | F40 | HL | OK | 0.975537 | 2.258473 | [0.11184678071289063, 0.1156472873046875, 0.11... | [-0.981106881873418, -0.9811217440723506, -0.9... | [78.60432842518559, 78.50442167364837, 78.4045... |
| 1 | 1179311 | F40 | VL | OK | 1.031965 | 2.345965 | [-0.16178969389648437, -0.1427871609375, -0.13... | [-1.0302631906068425, -1.0302753651956205, -1.... | [80.77476790838206, 80.67131586395837, 80.5678... |
| 2 | 1179312 | F40 | VL | OK | 0.928777 | 1.976692 | [-0.16178969389648437, -0.1427871609375, -0.13... | [-1.0302631906068425, -1.0302753651956205, -1.... | [80.77476790838206, 80.67131586395837, 80.5678... |

Figure 4.7: Cleaned data file

### 4.4.2 Spectrogram

As mentioned in sections above we have instances of acceleration velocity and position signal. The spectrograms are generated for each of this instances and sorted according to car model. The generated

graph have been stored in image format for analysis. The idea behind using this method is to gain knowledge of both spectral and temporal features and use it as a aid for classification. To generate spectrogram for signal was processed using Fourier analysis. The graph is generated using skit-learn library. Example of positive and negative sample spectrogram can be visualized in figure 4.8



a. Positive b. Negative

Figure 4.8: Example spectrogram of acceleration signal

# 5

# Solution

## 5.1 Hand crafted feature extraction

First approach to analyze and explore the domain knowledge of the raw signal from accelerometer is calculating temporal, spectral and statistical aspects of the data. As discussed in chapter 3 domain specific knowledge can be extracted from time-series data to analyze it and used as input to machine learning algorithms as raw signal cannot be used directly. Along with the calculated features we have prior information that has been extracted from the signal that is, penetration and maximum velocity. The dataset contains three signals acceleration, velocity and position that are passed to feature extractor which extracts fixed set of features.

| Type | Features |
|---|---|
| Statistical | Minimum, Maximum, Median, Mean absolute deviation,<br>Median absolute deviation,<br>Root mean square, Standard deviation,<br>Variance |
| Temporal | Mean absolute differences, Mean differences,<br>Median differences, Median absolute differences,<br>Entropy |
| Spectral | Minimum frequency, Maximum frequency,<br>Median frequency, Fundamental frequency |

Table 5.1: Fixed set of hand crafted features

The extractor function returns in total 54 features for each car door instance. Table 5.1 gives the overview of features calculated for each signal. The dimensionality of the feature vector obtained for each signal is reduced by removing the redundant features and selecting relevant ones by use of Random forest algorithm. This information is passed to supervised machine learning algorithm for classification.

## 5.2 Autoencoder for anomaly detection

To apply a general classification approach there is a requirement of fixed number of classes to separate the data. The problem that is solved by this project contains two classes normal and abnormal. Proper definition of what distinct feature that can classify the signals into these classes is missing. In other

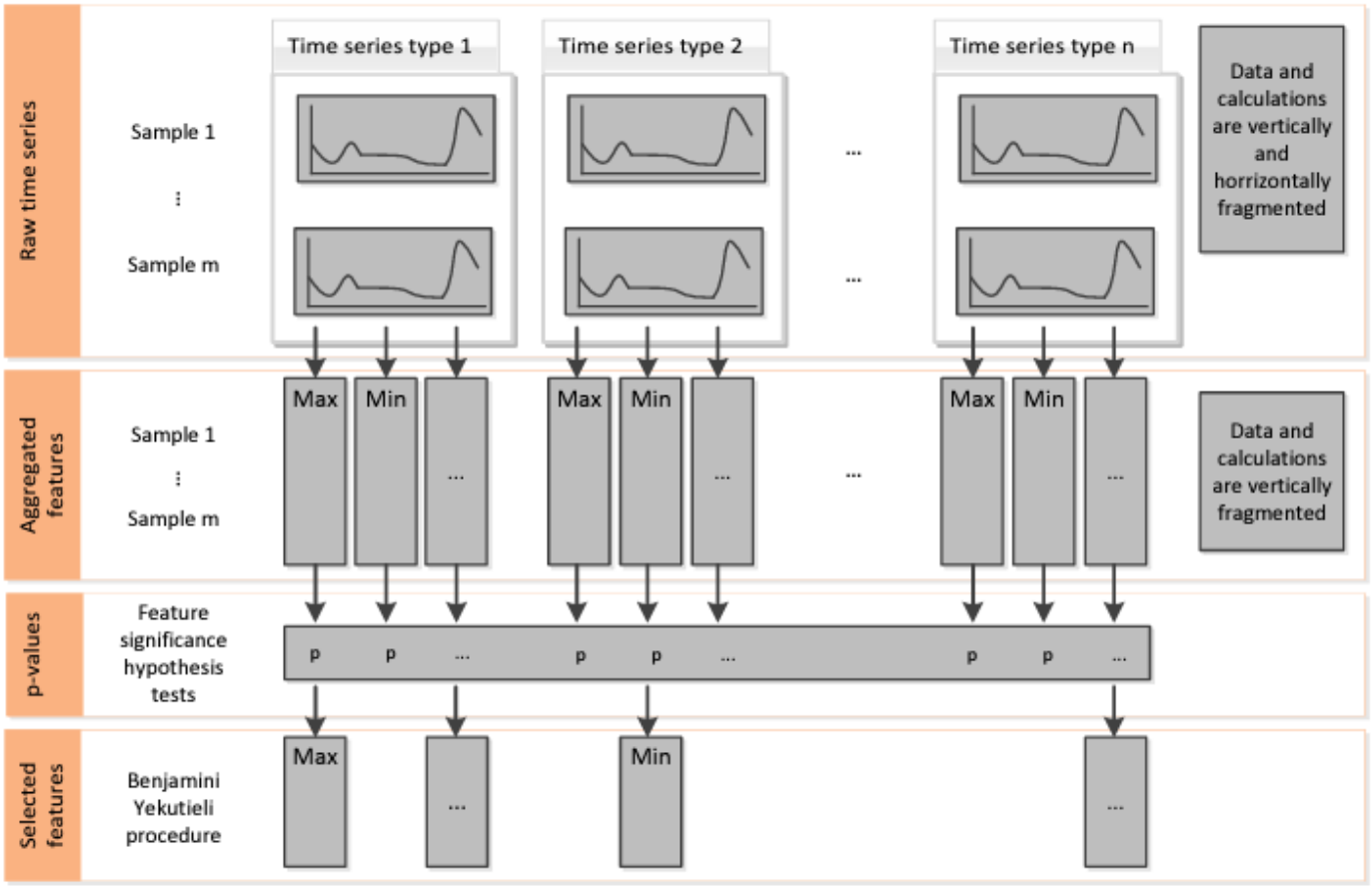



Figure 5.1: Process of extracting features from time-series data [11]

words prior information on to what can be called as anomaly is not available because the data we are dealing with is a complex industrial dataset. Also the data that is provided is highly imbalanced which has been seen during the data analysis in section 4.2.1. Therefore a direct implementation of deep-learning algorithm to this problem is difficult. These problems arise when we try to find a classification algorithm to solve the problem. To overcome these issues we propose to re frame the problem as anomaly detection using autoencoder.

**Algorithm 1:** Auto-encoder training algorithm.

**Input:** Sets of data to train $x_1, x_2 \dots x_n$
**Output:** encoder $f_\phi$, decoder $g_\theta$
    $\phi, \theta \leftarrow$ Initialize network parameters
**repeat**
    Compute mean squared error
    $L_{(\phi,\theta;x_i)} = \sum_i || x_i - g_\theta (f_\phi (x_i)) ||^2$
    $\phi, \theta \leftarrow$ Update parameters by SGD
**until** convergence of parameters $(\phi, \theta)$

Figure 5.2: Algorithm for anomaly detection [12]

Detailed discussion on general structure of autoencoder is done in chapter 3 section 3.1.3. The model is based on low-dimensional latent space which has high level abstraction of features. We do not require prior knowledge of latent variables in order to identify anomalies or outliers. Therefore as prior knowledge is not available for comfort door dataset general autoencoder is suitable for this problem.

The network has same number of inputs and outputs. The goal of this network is to minimize the reconstruction error between the decoded signal and the original. The problem now can be re-framed as regression rather than classification thus mean squared error is suitable for the problem to be solved.

During compression and expansion, the auto-encoder minimizes the loss of information. Although the decoder restores the high-frequency components to their maximum, the loss of the high-frequency components occurs during the compression process of the encoder. Signals are sampled for each element and reconstructed with the goal of reducing the average difference between input and output signals. The reconstructed signal is not equivalent to the original signal.Instead of focusing on if the input is normal or abnormal the reconstruction error is considered. The algorithm learns higher abstraction of data because of dimension reduction.

In the residual error based anomaly detection algorithm computation of the threshold is done based on the validation set. A optimal threshold can be selected if the it is computed based on abnormal data. A trained auto-encoder will predict the unseen normal data with small reconstruction error as the distribution of samples is similar. When it predict anomalous data the reconstruction error is high. Thus reconstruction error acts a aid to catch the rare events. Figure A.1 shows the architecture of autoencoder

**Algorithm 2:** Residual error-based anomaly detection algorithm.

**Input:** Sets of data to classify $x_1, x_2 \ldots x_n$
**Output:** Residual error $L_{(x,\hat{x})}$
$\phi, \theta \leftarrow$ train network parameters with Algorithm 1
$\alpha \leftarrow$ set threshold based on training set
**repeat**
**for** i=1 to N **do**
Compute residual error $L_{(x,\hat{x})}$
$L_{(\phi,\theta;x_i)} = \sum_i || x_i - g_\theta (f_\phi (x_i)) ||^2$
**if** residual error $L_{(x,\hat{x})} > \alpha$ **then**
$x_i$ is abnormal
**else**
$x_i$ is normal
**end if**
**end for**

Figure 5.3: Algorithm for anomaly detection [12]

model used.

## 5.3 Transfer learning with raw signal data

There are very few open source resources trained model which has similar data structure to the complex industrial dataset used for this project. As discussed in the state of art the most popularly used transfer learning model for time series data is CNNs. Many variations of the this model can be found. The model which is used deep convolution network trained with data shown in figure A.2. This architecture is chosen based from the paper [41]

The steps that are followed for use of transfer learning using the above model from raw time series data are as follows:

1. As it can be seen in the figure 5.4 the model is copied along with its weights.
2. The acceleration signal from time series data is resized according to the model input requirements.
3. First few layers of the model are frozen.
4. The frozen part acts as feature extractor and this is classified by the model parameters that are trainable.

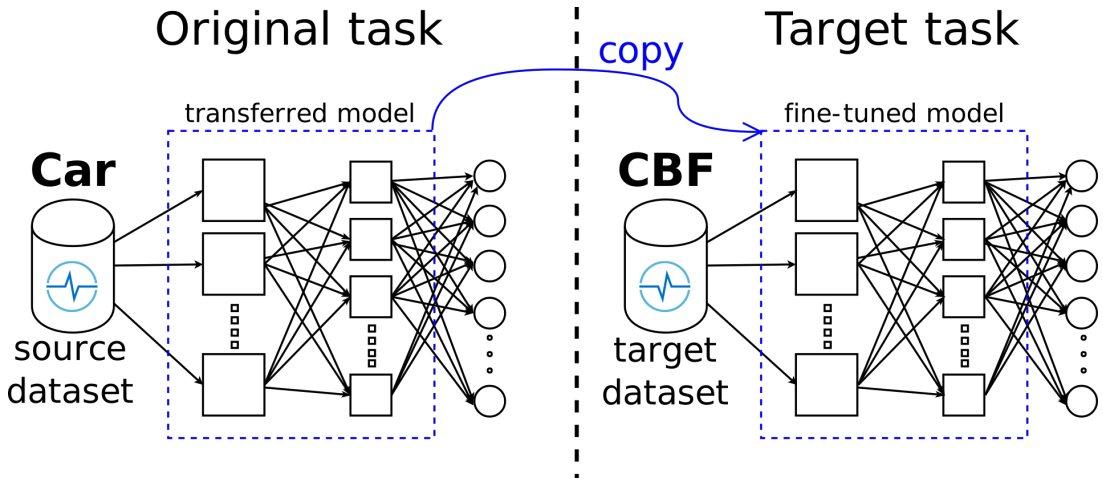



Figure 5.4: Transfer learning model [12]

5. Instead of softmax sigmoid function has been used due to presence of only two classes for the problem.

## 5.4 Transfer learning with spectrogram

There is a significant work done to designing networks to classify and extracting features from spectrogram. Finding a suitable deep-learning model when we have low instances of data is difficult. Therefore transfer learning is helpful solving this problem. Various popular networks architectures such as VGG and Resnet have been used to extract features from spectogram data. For this project VGG is used.

VGG which is a image classification model and has no relation to the temporal domain knowledge in its design. Also as discussed earlier a clear definition to anomaly is missing. Even with these issues VGG is suitable for this problem because, not considering the domain knowledge minimizes the assumptions that are done with respect to the specific domain. Deep learning models are known to be perming better if the feature space is not constrained with domain knowledge. Rather than make assumptions about the type of signal or problem, VGGs combine small-context representations all hierarchically, so that any structure can be learned [42].

The steps that are followed for use of transfer learning from spectrogram are as follows:

1. First step is to pre-process the image in such a manner that it is in suitable format of input for VGG network. Required input shape for the network is $(244, 244, 3)$. Also the data should be normalized to values 0 to 1 by dividing the array with factor of 225.

2. Next step is to load VGG model with pre-trained weights for imgenet dataset.

3. The feature extractor part of the network is defined from the network input to the last max pooling layer (labeled by 7 x 7 x 512). The remaining part is regarded as the classification part.The feature extraction part of the model is frozen to ensure the weights of the network do not change during model training.

4. To scale and convert the image into Numpy array module from Keras [43] library has been used.
5. The classification part of the network is trained with the features obtained from the extractor part of the network.
6. Instead of softmax sigmoid function has been used due to presence of only two classes for the problem.

## 5.5 Autoencoder as feature extractor

Auto-encoder are popularly used in computer vision algorithms as feature extractor of images. Many models have been successfully implemented in this field. Here through this implementation similar approach is being experimented with spectrogram as input. This method might be useful to extract some consized features from the image data as using the them directly will increase the training complexity and time.

The implementation details are as follows:

1. First simple preprocessing is done by scaling the image data and converted to numpy array.
2. Then the data is trained on to the auto-encoder by minimizing the reconstruction error.
3. Features for training and test are extracted from the encoder predictions.
4. Further this features are used as input to a simple convolution network with the aim of classification.

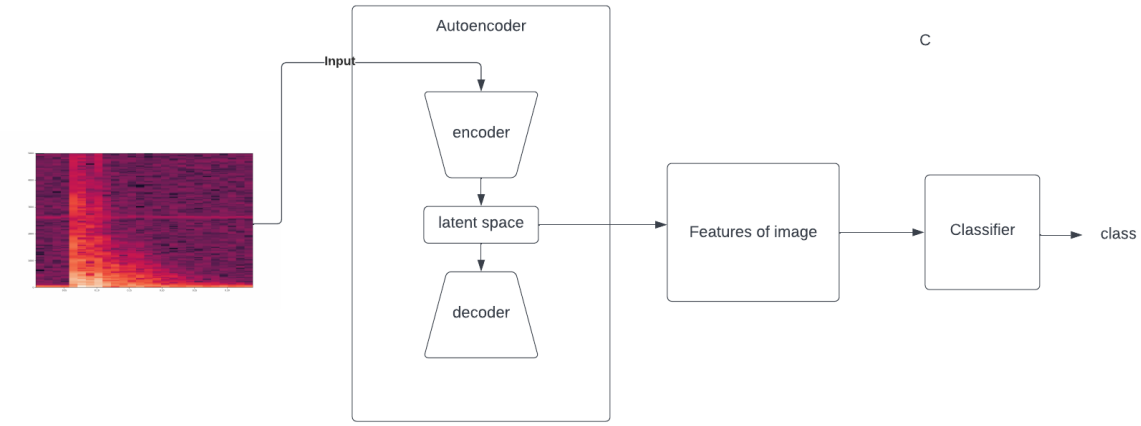



Figure 5.5: Pip-line of implementation of autoencoder as feature extractor

# 6

# Hand crafted feature extraction

This chapter discusses the experiments conducted based on methods present in chapter 5 section 5.1.

### 6.0.1 Objective

Objective of this experiment is to extract a fixed set of features from each of the signal acceleration, velocity and position. From this fixed set relevant features will be selected and classified using state of art machine learning algorithms.

## 6.1 Experiment details

As the data is imbalanced therefore to perform data balancing following set of data division is performed. Following table describes data balancing experiments performed.

| Data division | Details | Positive samples | Negative samples |
|---|---|---|---|
| D1 | The data is divided in 1:1 ratio. | 103 | 103 |
| D2 | The data is slightly biased 2:1 | 206 | 103 |

Table 6.1: Data division for balancing

The number of features selected varies as follows:

- First feature set is defined as feature selection based on relevance let us call it as F1.
- To analyze the effect of the prior domain knowledge features, set of two is selected that is penetration and maximum velocity. Let us call this F2.

## 6.2 Results

The relevance of the features can be visualized as follows: Results related to data division D1 and D2 category are presented as in figure 6.2 and 6.3: Parameters related to the Testing data:

- Positive samples: 698
- Negative Samples:38

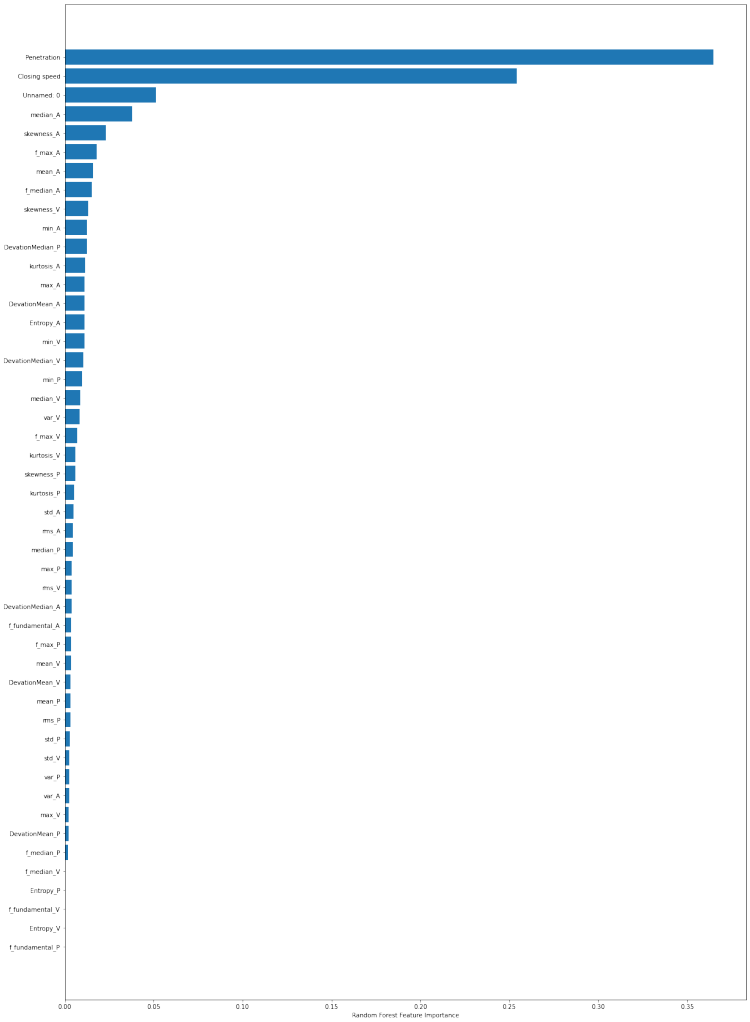
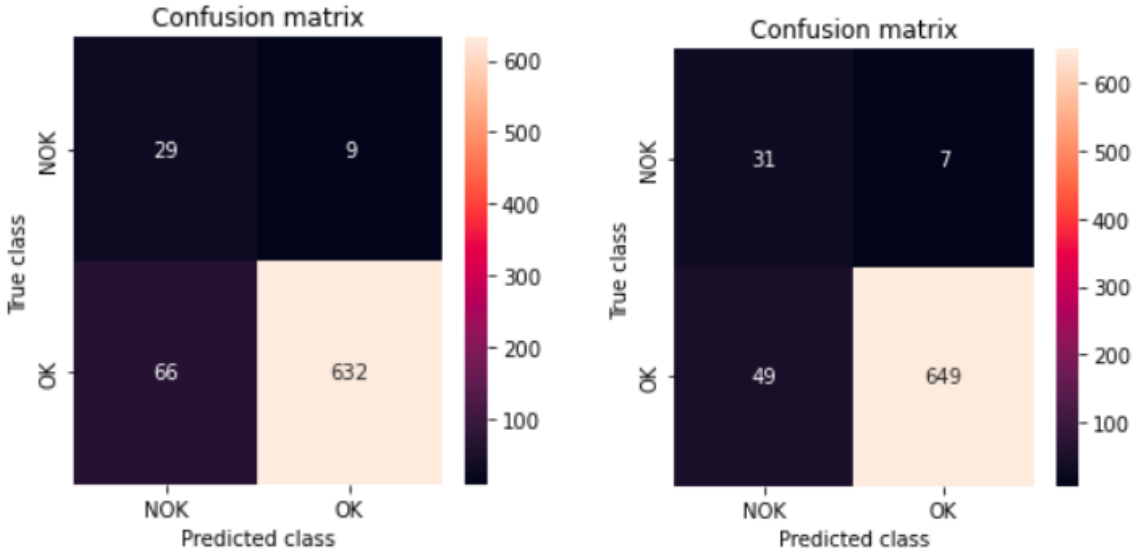



Figure 6.1: Relevance of features

## 6.3 Discussions

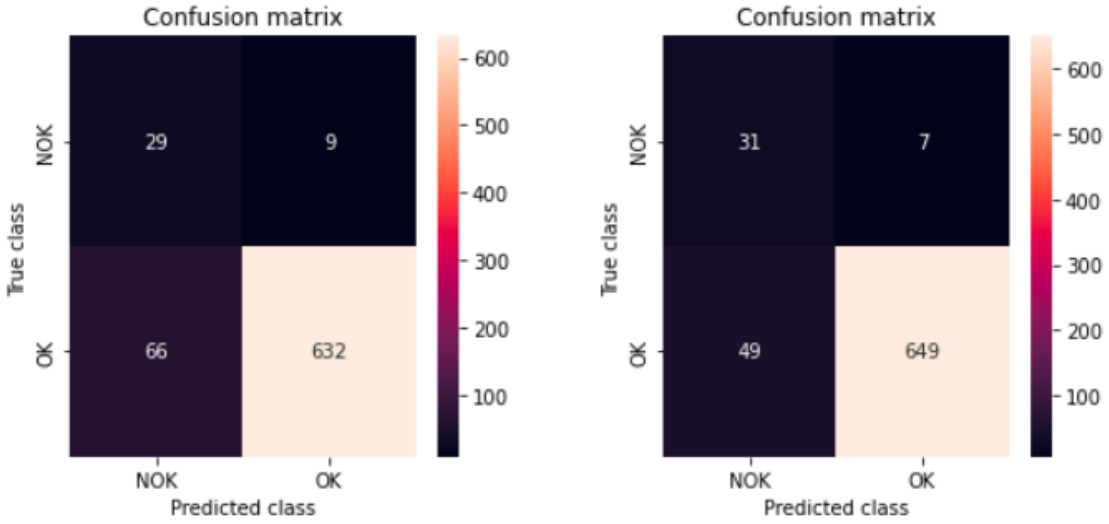
As observed the algorithm performs better for when the prior knowledge is used that is the F2 category. Both the categories show no significant difference in number of negative samples obtained by the algorithms. This shows that the prior domain knowledge has high influence on negative sample detection.




a. F1(Accuracy:74.72%)

b. F2(Accuracy: 86.4%)

Figure 6.2: Results for D1 category




a. F1(Accuracy:89.8%)

b. F2(Accuracy: 92.3%)

Figure 6.3: Results for D2 category

# 7

# Autoencoder for anomaly detection

This chapter discusses the experiments conducted based on methods present in chapter 5 section 5.2.

## 7.1 Objective

Aim of this experiment is to use reconstruction error as base to classify the data into normal and abnormal. This experiment does not consider the labels assigned that is, its a unsupervised learning algorithm. The details of architecture have been discussed in section 5.2. This experiment is conducted on each type of signal to analyze which signal has better features that encode the anomaly.

## 7.2 Results

### 7.2.1 Trained with positive data:T1

Parameters related to the data distribution:

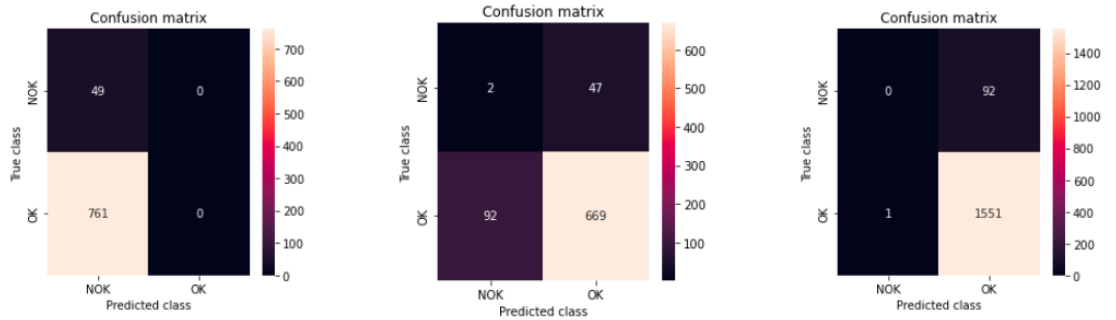
| Data | Positive samples | Negative samples |
|---|---|---|
| Train set | 2311 | 0 |
| Test set | 1848 | 112 |

Table 7.1: Parameters for T1

### 7.2.2 Trained with combination of positive and negative data: T2

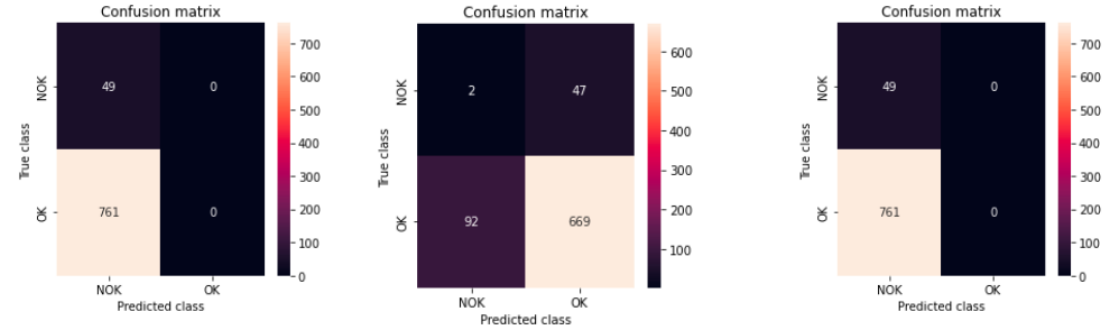
| Data | Positive samples | Negative samples |
|---|---|---|
| Train set | 1550 | 761 |
| Test set | 1848 | 34 |

Table 7.2: Parameters for T2




a. Acceleration signal(Accuracy:6.2%) b. Velocity signal(Accuracy: 82.3%) c. Position signal(Accuracy: 94.3%)

Figure 7.1: Results for T1




a. Acceleration signal(Accuracy:6.2%) b. Velocity signal(Accuracy: 82.6%) c. Position signal(Accuracy: 6.43%)

Figure 7.2: Results for T2

## 7.3 Discussions

As it can be visualized from the 7.1 and 7.3 even though the accuracy of the experiments for velocity is high it fails to detect abnormal data. The reason behind this could due to the reconstruction error of the anomalies is very close to normal ones. This can be visualized in figure 7.3. Optimal threshold cannot be decided due to this issue as if we increase the threshold most of the normal data will be classified as abnormal. Therefore the threshold is kept very low and it can be observed in the confusion plots that most of abnormal data is been classified as normal.

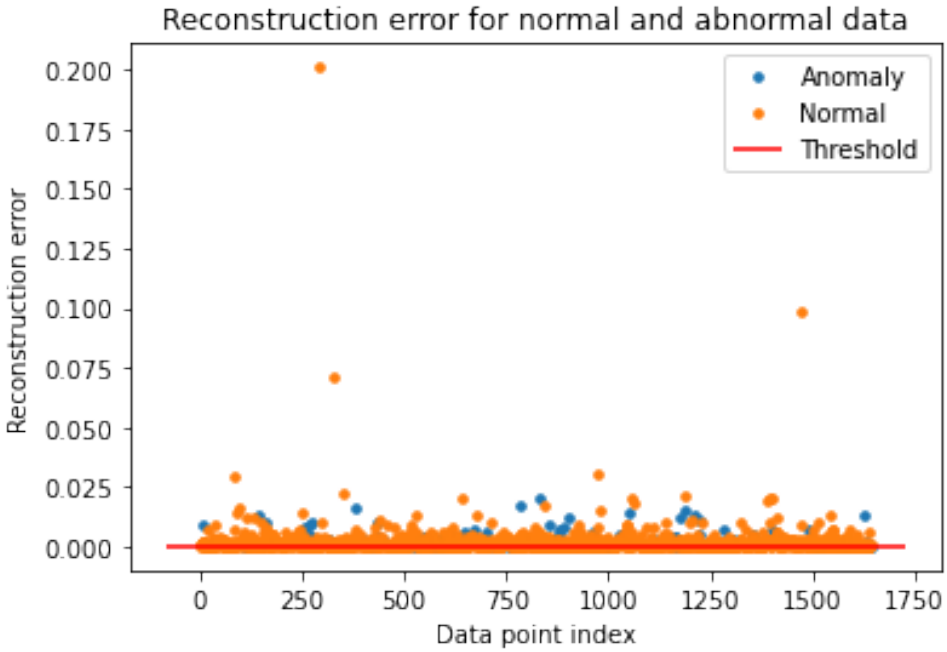



Figure 7.3: Reconstruction error distribution

# 8

# Transfer learning

## 8.1 Signal as input

### 8.1.1 Objective

Objective of this experiment is to use convolution neural network pre-trained with UCR datasets as base model for feature extraction from comfort door acceleration data. The results

# 9

# Conclusions

**9.1 Contributions**

**9.2 Lessons learned**

**9.3 Future work**

# A

# Design Details

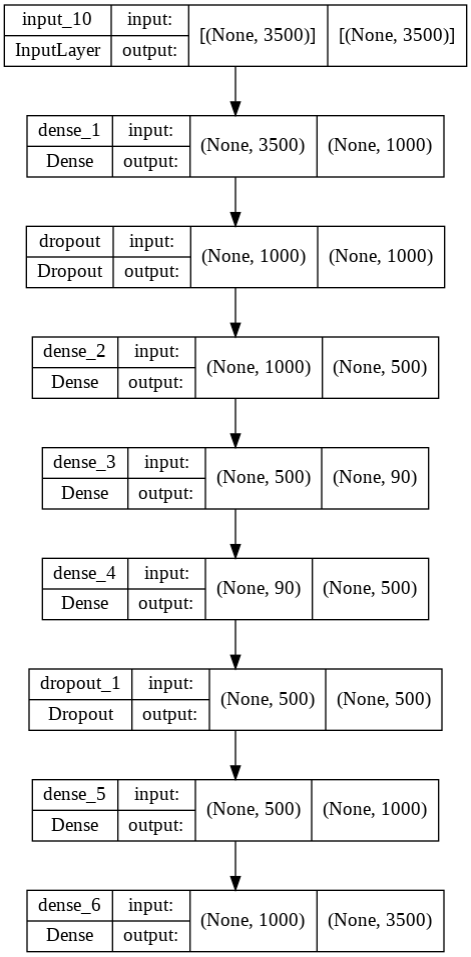



Figure A.1: Autoencoder architecture for anomaly detection

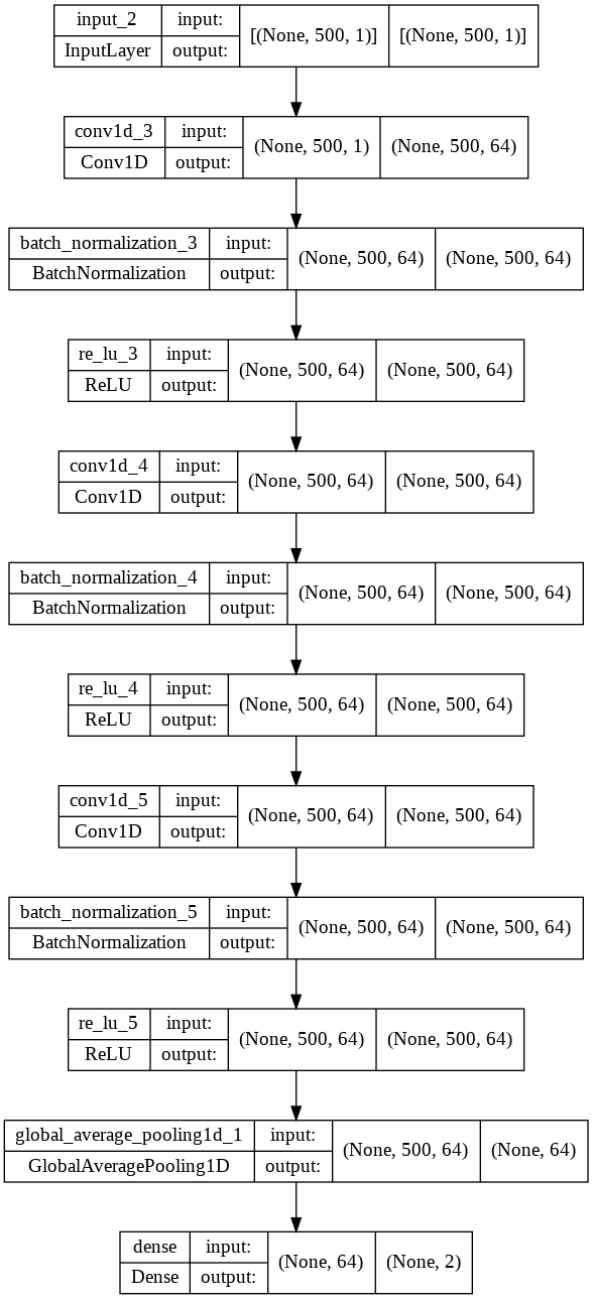



Figure A.2: CNN architecture for transfer learning

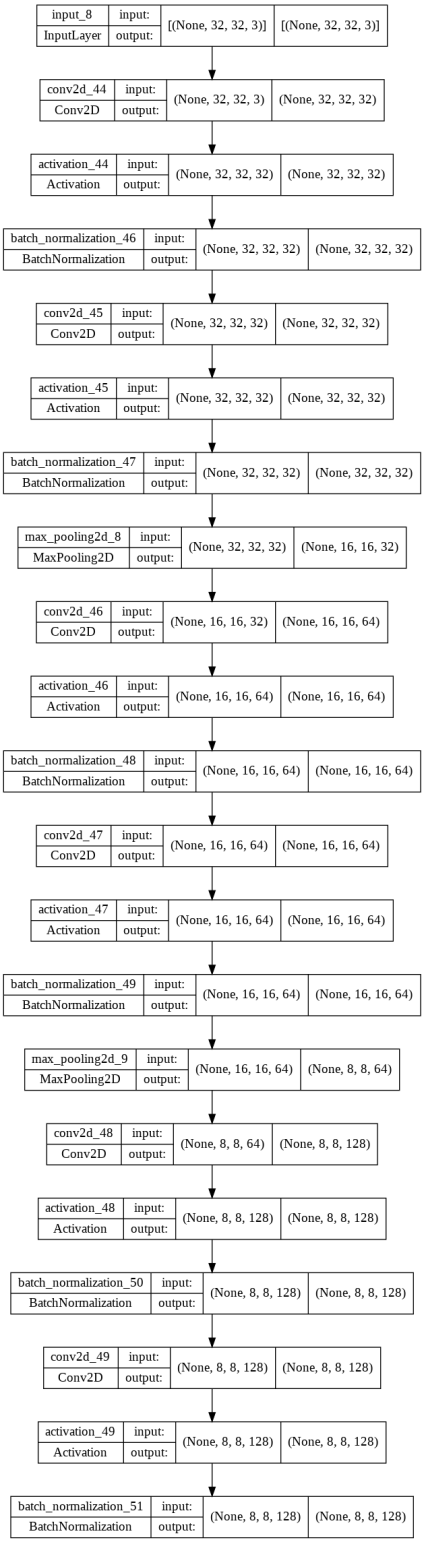



Figure A.3: Encoder architecture

# References


[1] H. F. Nweke, Y. W. Teh, M. A. Al-Garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Systems with Applications*, vol. 105, pp. 233–261, 2018.

[2] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[3] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," in *ECML/PKDD workshop on advanced analytics and learning on temporal data*, 2016.

[4] "Deconstructing time series using fourier transform," https://medium.com/@khairulomar/deconstructing-time-series-using-fourier-transform-e52dd535a44e, accessed: 2022-02-02.

[5] "Plotting a spectrogram using python and matplotlib," https://pythontic.com/visualization/signals/spectrogram, accessed: 2022-02-02.

[6] "Applied deep learning - part 3: Autoencoders," https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798, accessed: 2022-02-02.

[7] "What is the convolutional neural network architecture?" https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/, accessed: 2022-02-02.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] H. Choi, S. Ryu, and H. Kim, "Short-term load forecasting based on resnet and lstm," 10 2018, pp. 1–6.

[11] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial big data applications," *arXiv preprint arXiv:1610.07717*, 2016.

[12] D. Y. Oh and I. D. Yun, "Residual error based anomaly detection using auto-encoder in smd machine sound," *Sensors*, vol. 18, no. 5, p. 1308, 2018.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[14] C. Herff and D. J. Krusienski, "Extracting features from time series," *Fundamentals of clinical data science*, pp. 85–100, 2019.

[15] M. Murugappan and S. Murugappan, "Human emotion recognition through short time electroencephalogram (eeg) signals using fast fourier transform (fft)," in *2013 IEEE 9th International Colloquium on Signal Processing and its Applications*. IEEE, 2013, pp. 289–294.

[16] N. Saravanan and K. Ramachandran, "Incipient gear box fault diagnosis using discrete wavelet transform (dwt) for feature extraction and classification using artificial neural network (ann)," *Expert systems with applications*, vol. 37, no. 6, pp. 4168–4181, 2010.

[17] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[18] A. Supratak, L. Li, and Y. Guo, "Feature extraction with stacked autoencoders for epileptic seizure detection," in *2014 36th Annual international conference of the IEEE engineering in medicine and biology society*. IEEE, 2014, pp. 4184–4187.

[19] R. H. Shumway, D. S. Stoffer, and D. S. Stoffer, *Time series analysis and its applications*. Springer, 2000, vol. 3.

[20] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," *International Journal of Computer Research*, vol. 10, no. 3, pp. 49–61, 2001.

[21] P. Geurts, "Pattern extraction for time series classification," in *European conference on principles of data mining and knowledge discovery*. Springer, 2001, pp. 115–127.

[22] T. Lambrou, P. Kudumakis, R. Speller, M. Sandler, and A. Linney, "Classification of audio signals using statistical features on time and wavelet transform domains," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, vol. 6. IEEE, 1998, pp. 3621–3624.

[23] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 12, pp. 3026–3037, 2014.

[24] A. De Brabandere, P. Robberechts, T. Op De Beéck, and J. Davis, "Automating feature construction for multi-view time series data," in *ECMLPKDD Workshop on Automating Data Science*. N/A, 2019, pp. 1–16.

[25] S. Gowid, R. Dixon, and S. Ghani, "A novel robust automated fft-based segmentation and features selection algorithm for acoustic emission condition based monitoring systems," *Applied Acoustics*, vol. 88, pp. 66–74, 2015.

[26] T. W. Joo and S. B. Kim, "Time series forecasting based on wavelet filtering," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3868–3874, 2015.

[27] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[28] V. Kuznetsov, V. Moskalenko, and N. Y. Zolotykh, "Electrocardiogram generation and feature extraction using a variational autoencoder," *arXiv preprint arXiv:2002.00254*, 2020.

[29] S. Mehtab and J. Sen, "Analysis and forecasting of financial time series using cnn and lstm-based deep learning models," in *Advances in Distributed Computing and Machine Learning*. Springer, 2022, pp. 405–423.

[30] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems," *Scientific reports*, vol. 9, no. 1, pp. 1–16, 2019.

[31] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[32] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[33] Y. Ren, J. Mao, Y. Liu, and Y. Li, "A novel dbn model for time series forecasting," *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 79–86, 2017.

[34] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "A review of human activity recognition methods," *Frontiers in Robotics and AI*, vol. 2, p. 28, 2015.

[35] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[36] K. P. Thanaraj, B. Parvathavarthini, U. J. Tanik, V. Rajinikanth, S. Kadry, and K. Kamalanand, "Implementation of deep neural networks to classify eeg signals using gramian angular summation field for epilepsy diagnosis," *arXiv preprint arXiv:2003.04534*, 2020.

[37] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[38] "What are the spectral and temporal features in speech signal?" https://www.researchgate.net/post/What-are-the-Spectral-and-Temporal-Features-in-Speech-signal/54fb90d1d11b8b897b8b4567/citation/download., accessed: 2022-02-02.

[39] B. Gerazov, G. Bailly, O. Mohammed, Y. Xu, and P. N. Garner, "A variational prosody model for mapping the context-sensitive variation of functional prosodic prototypes," *arXiv preprint arXiv:1806.08685*, 2018.

[40] D. Ravi, N. Ghavami, D. C. Alexander, and A. Ianus, "Current applications and future promises of machine learning in diffusion mri," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 105–121.

[41] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Transfer learning for time series classification," in *2018 IEEE international conference on big data (Big Data)*. IEEE, 2018, pp. 1367–1376.

[42] "Why do spectrogram-based vggs rock?" https://towardsdatascience.com/why-do-spectrogram-based-vggs-rock-6c533ec0235c, accessed: 2022-02-02.

[43] N. Ketkar, "Introduction to keras," in *Deep learning with Python*. Springer, 2017, pp. 97–111.