A Machine learning based approach to identify freezing-of-gait in Parkinson's Disease patients using recurrence plots

Aniketh Yadav

Supervisor: Dr. Mark Hoogendoorn

Associate Professor

Faculty of Sciences

VU University Amsterdam

Master thesis submitted in fulfillment

of the requirements for the degree in

Master of Science in Computer Science

Academic year *2019-2020*

Abstract

Feature engineering and outlier treatment are among the most critical and time-consuming steps in any data analysis process. The substantial amount of research work in Parkinson's Disease (PD) identification has culminated with the generation of complex and arduous handcrafted feature engineering techniques. Apart from being complicated and arduous, the handcrafted feature engineering techniques are often susceptible to noise and outliers. The complex and arduous nature of handcrafted feature engineering techniques obstructs a timely diagnosis of PD. Timely diagnosis of PD is essential because the disease progresses quickly and worsens the motor and non-motor related difficulties. The authors propose a recurrence plot-based convolutional neural network (RPCNN), which negates the requirement of handcrafted feature engineering and dedicated outlier treatment. The evaluation of the RPCNN architecture shows promising results towards the development of a fully automated PD identification technique which is robust against the presence of outliers.

Contents

Chapter 1

Introduction

Over the past few years, we have seen enormous leaps in the ability to store, analyse and process data. Computers have become smaller and faster, and with services such as amazon web services, we can opt for cloud computing platforms on a Pay-As-You-Go basis. A combination of all such factors has made it possible for us to generate and store massive amounts of data[52]. The two crucial steps in the life cycle of any data analysis project are data pre-processing and feature engineering[54]. The real-world data, in its raw form, is often incomplete and inconsistent. Hence, we need to "clean" the data before we analyse it. This step of cleaning the data to make it ready for analysis is often referred to as *data pre-processing*. The nature of data pre-processing depends on the type of data and the context of data analysis problem. If the data is continuous (for example, the data taken from sensors is continuous data), then data de-noising and outlier treatment are of paramount importance. After pre-processing, feature engineering is another crucial step wherein the data is transformed in such a way that it helps in the analysis of data[5]. Feature engineering techniques are generally performed under the supervision of domain experts[10]. Hence, many feature engineering techniques involve the laborious design of complex algorithms which are used only for analysis in a particular domain.

In the detection of Parkinson's disease (PD), various motor and non-motor related symptoms are identified to classify a subject as either a patient with Parkinson's disease (PWP) or a healthy control (HC). A literature survey conducted by the authors reveals that *data pre-processing* and *feature extraction* are the two crucial steps in designing any computer-aided PD identification system. In these PD identification systems, data pre-processing is either a part of the feature extraction or specific *data cleaning* is performed. In most of the PD identification research works, a thorough analysis and understanding of the data are followed by manual extraction of features from the data. The design and implementation of such *handcrafted feature engineering* processes are laborious, complex and often susceptible to noise and outliers in the data. In the case of PD identification, the timely diagnosis is essential to prevent the rapid progression of the disease. For timely diagnosis of PD, the time-consuming and noise-susceptible nature of handcrafted feature engineering processes is a huge challenge.

Recently, machine learning techniques like artificial neural networks (ANNs) and its variants like the convolutional neural networks (CNNs) are utilized to negate the laborious task of manually extracting features in PD identification[1][24] [12][56][40]. Even though the deep learning techniques are automating feature extraction, the data is often subjected to de-noising and outlier treatment. To reduce the dependency on *handcrafted feature engineering* and *explicit outlier treatment*, the authors propose and evaluate a recurrence plot (RP) based CNN classifier, thereby referred to as RPCNN. Afonso et al. also implemented a similar approach to identify PWP from the data generated via a smart pen[37]. The research work of Afonso et al. is extended by evaluating the feasibility and performance of the proposed RPCNN architecture. The authors evaluate the following aspects of the RPCNN: *robustness against noisy data*, *performance against benchmarks* and *generalizing power on unseen data*. The evaluation of the RPCNN architecture shows promising results in developing a PD identification system which reduces the dependency on handcrafted feature engineering and specific outlier treatment.

In the following chapter, we briefly give a background on PD and proceed to explain how PD is detected by identifying the various symptoms. We also touch upon the architectures of RPs and CNNs. We illustrate how RPs negate the requirement of outlier treatment and how CNNs automatically identifying features for classification.

Chapter 2

Background

To understand how the proposed RPCNN architecture works and why such an architecture is required in PD detection, a basic background understanding is required. In this chapter, the authors describe the background to the research problem by addressing two key components:

1. Background on the PD domain: This section gives a brief understanding of PD, why research work in PD detection is required and the current challenges of computer-aided PD detection.

2. Background on the algorithm domain: As the proposed architecture involves the use of RPs and CNNs, a brief background on RPs and CNNs is included.

2.1 Parkinson’s Disease

2.1.1 Introduction

Parkinson’s disease (PD) falls under the spectrum of neurodegenerative diseases, and it is the second most common condition of this category, after Alzheimer’s. Despite its prevalence worldwide, diagnosing PD is still a challenge, particularly in the earliest stages, and it relies heavily on clinical evidence[8].

2.1.2 Why is research in PD important?

PD is sporadic among people with age less than 50, but as the age increases, the number of people affected increases rapidly. More than 1% of the population over the age of 60 and approximately 4% at the age of 80 years are affected by it, making it the most prevalent movement disorder and the second most common neurodegenerative disorder after Alzheimer’s disease [2]. Due to the development of medical science and technology, the number of people aged more than 50 years is growing all around the globe. According to [51] the number of people aged 60 or older will be higher than the number of children aged less than five years, this will impose a growing financial and social burden on industrialised, ageing populations. Hence there is an increasing need to develop health care, which targets the ageing population of the world who are susceptible to Parkinson’s disease.

Neuropathically, PD is defined by loss of dopaminergic neurons in the substantia nigra pars compacta of the midbrain. Another important neuropathology is Lewy bodies. Lewy bodies refer to the abnormal accumulation of alpha-synuclein in the cytoplasm of specific neurons in several different brain regions[19]. Lewy bodies were the first to be described over a century ago. Patients with Alzheimer's disease also have Lewy bodies, but these Lewy bodies are concentrated mainly in limbic brain regions[21].

2.1.3 Symptoms of PD

Two broad classifications of PD include motor and non-motor symptoms. Motor symptoms of PD refer to the inability to perform a bodily movement properly. These include[39]:

Tremor/dyskinesia - a shaking of the hands, arms or legs, especially when the limb is at rest.

Rigidity - abnormal stiffness in a limb or part of the body.

Postural instability - impaired balance or difficulty standing or walking.

Bradykineasia - gradual loss and slowing down of spontaneous movement. Though many of the symptoms indicate towards PD being a motor disorder it has a multitude of non-motor related disorders such as cognitive impairment, autonomic dysfunction, disorders of sleep, depression and hyposmia, are part of the disease and add significantly to overall burden.

Freezing-of-gait (FOG)/Akinesia - PD patients experience sudden stop in motion while performing ADL (activities of daily living), this is called freezing-of-gait.

2.1.4 Challenges in diagnosis of PD

Despite the advances in neuroimaging and genetics, the primary diagnosis of PD remains clinical. A definite diagnosis is only obtained pathologically[11]. All the current diagnostic criteria consider signs and symptoms emerging in the later stage of the disease. By the time-definite diagnosis is reached, the disease has advanced in the body.

PD can be challenging to accurately diagnose, particularly in the early stages of the disease, which is why a neurologist trained in movement disorders is critical. The clinical presentation of Parkinson's disease is heterogeneous and overlaps with other conditions, including the parkinsonian variant of multiple system atrophy (MSA-P), progressive supranuclear palsy (PSP) and essential tremor.‘ A study[35] found that, there was a misdiagnosis rate of 24% and that this strongly depended on who was performing the diagnosis and whether or not they were applying diagnostic criteria based on clinical guidelines. Specialists who were not movement disorder experts had a correct diagnosis rate of only 75% and diagnosed by primary care doctors had a correct diagnosis of just 53%. Hence, there is a need to research and develop new methods which accurately and effectively diagnose PD.

2.1.5 Computer-aided diagnosis of PD

A substantial amount of research work has been conducted to identify and diagnose PD with the help of computers. PD is diagnosed by identifying the various motor and non-motor related symptoms of the disease. For computer-aided detection, quantifying and capturing the motor-related symptoms is easier than non-motor symptoms. Depending on the nature of the symptom, different technologies are utilised to capture specific data. For example, tremor in the voice, which corresponds to dyskinesia, is captured using microphones[29]. FOG, which corresponds to akinesia, is captured by either motion sensors or by analysing the image/video data of the patients[7]. The tremor in hands, which also corresponds to dyskinesia, is captured by placing motion sensors on the arms and wrists of the patients[37].

2.1.6 Challenges in computer-aided diagnosis of PD

As the PD detection algorithms focus on identifying particular symptoms, the design and implementation prove to be complex and time-consuming. Despite being complex and time-consuming, the PD detection algorithms are not only sensitive to the presence of outliers but also rely heavily on the complex handcrafted feature engineering. The complex and time-consuming nature of PD detection algorithms is a hindrance for any real-time application, as timely diagnosis of PD is important[35]. Delay in PD diagnosis results in the rapid progression of the disease with worsening of the symptoms. Hence, there is a need to develop PD identification systems which are not hindered by complex and time-consuming handcrafted feature engineering.

2.1.7 Automatic feature engineering and noise proof identification systems

To address the issue of latency caused by the complex and arduous nature of handcrafted feature engineering, researchers are focusing on various machine learning approaches. ML algorithms, like artificial neural networks (ANNs), have the ability to detect and populate features automatically. Hence, by automating the feature engineering approach, the bottleneck in the form of complex handcrafted feature engineering processes is removed. Recently, ANN models [40][20][44][33][23], ANN in conjunction with hidden markov models (HMMs) [46], 1-dimensional CNNs [56][12] have been utilized to achieve the goal of removing the dependency on complex handcrafted feature engineering. Even though the models mentioned above automate feature engineering, they are still sensitive to the presence of outliers. Hence, outlier treatment and noise removal is an integral part of any PD identification system. The only paper which addresses the goal of automatic feature engineering on noisy data is a recently published research paper by Afonso et al. [37]. Pereira et al. achieved automatic feature engineering on raw signals using RPs in conjunction with CNNs. The authors extend the research work of Afonso et al. by creating new RPCNN architecture and subjecting it to the various evaluation methods mentioned in chapter 5.

2.2 Machine learning and Artificial Neural Networks

Machine learning algorithms such as artificial neural networks are increasingly used to automate feature engineering processes. Hence, it is helpful to have a basic understanding of machine learning and artificial neural networks.

2.2.1 Machine learning

Machine learning comes under the umbrella of artificial intelligence (AI), in which the algorithms are *data-driven*. The driving force behind ML is not just the algorithm itself, but also the data available at hand[10].

2.2.2 Types of Machine learning

There are three major approaches[10] in the field of ML: supervised, unsupervised, and reinforcement learning.

1. **Supervised learning**

In supervised ML algorithms, the input data consists of both input data and the desired outputs. The outputs can be different categories (also called "labels") or continuous values. If the data consists of categories, then the Ml algorithm is called a **classification** algorithm. In classification, the ML algorithm learns to classify the given data into different categories based upon the observation of different patterns. When the desired output consists of a range of numbers instead of categories, then the ML algorithm is called a **regression** problem.

2. **Unsupervised learning**

In unsupervised learning, the input data does not contain any "labels" or output. Unsupervised learning algorithms find patterns in the data by grouping "similar" data points together. The data points are grouped in correlation to their "similarity". Different metrics are utilised to define similarity. The measure of similarity is generally the distance between the data points, for instance: euclidean, manhattan, mahalanobis distance[10]. The value of the distance metric corresponds to the "similarity" between the data points.

3. **Reinforcement learning**

In reinforcement learning, there are three entities: an *agent*, an *environment* in which the agent exists and a *feedback*. The feedback is either a "reward" (positive reinforcement) or a "penalty" (negative reinforcement). Based upon the feedback received, the agent changes its actions through the environment.

2.2.3 Artificial neural networks

Artificial neural networks (ANNs) are a type of ML algorithm, which takes inspiration from the structure of the human neurons. To understand how an ANN automatically extracts features from the data, we need to understand the basic working of a vanilla ANN. The current mathematical models of ANNs draw inspiration from the works of [43].

2.2.4 Single neuron

To understand how the entire ANN works, we will start by understanding the working of a single neuron. The 4 terms associated with a single neuron are:

1. Inputs: represented by *x1, x2,...., xN*
2. Weights: represented by *w1, w2,...., wN*
3. Bias: represented by *b*
4. Output: represented by *a*

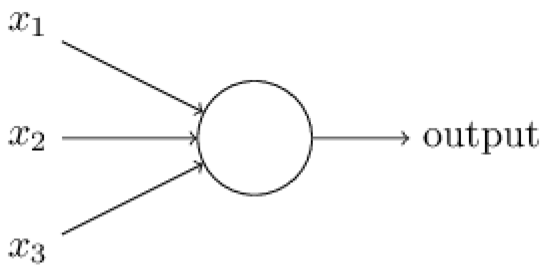


Figure 2.1: Diagram of a single neuron.

A single neuron takes inputs *x1, x2,...., xN*, add the weighted sum of the inputs with a bias *w1, w2 ..., wN* and produces an output *a*. The output *a* is generated by applying an activation function to the sum of weighted inputs with the bias. Here we represent this activation function with *f*. Mathematically this is represented as:

$$a = f(x_1w_1 + x_2w_2..... + b) \tag{2.1}$$

Using the summation notation, we end up with with the following concise equation:

$$a = f(\sum_{i=1}^{N} x_\mathrm{i}w_\mathrm{i} + b) \tag{2.2}$$

2.2.5 Multiple "Layers" of neurons

To extend the equation to multiple neurons in multiple layers, we introduce new notations. We will use ${w_{jk}}^l$ to denote the weight for the connection from the k^{th}

neuron in the *(l −1)*th layer to the *j*th neuron in the *l*th layer.

So, for example, the diagram below shows the weight on a connection from the *4*th neuron in the *2*nd layer to the *2*^{nf} neuron in the *3*rd layer of a network:

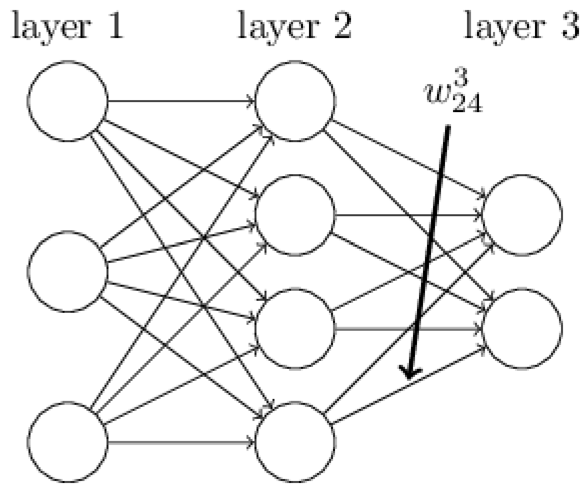


Figure 2.2: Notation of weights.

We use a similar notation for the network's biases and activations. Explicitly, we use $b_j{}^l$ for the bias of the *j*th neuron in the *l*th layer. And we use $a_j{}^l$ for the activation of the *j*th neuron in the *l*th layer. The following diagram shows examples of these notations in use:

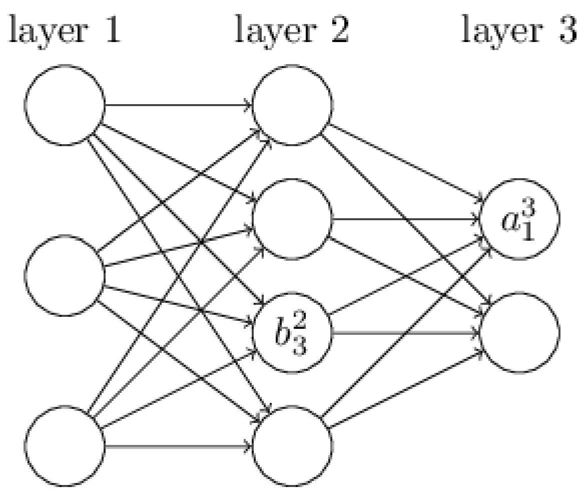


Figure 2.3: Notation of activation outputs and biases.

With these notations, the activation $a_j{}^l$ of the j^{th} neuron in the l^{th} layer is related to the activations in the $(l-1)^{th}$ layer by the equation

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) \tag{2.3}$$

Therefore, for a given set of weights and biases, we can iteratively calculate the final output of a neural network. Once the output is calculated, we quantify the difference between the *predicted output value* (final output from the final layer) and the *actual output value* (the value present in the dataset). A *cost function* achieves the comparison between the predictions and actual values.

2.2.6 Cost function

Cost function quantifies the difference between the *predicted output* of the neural network against the *actual output* in the data. The nature of the cost function depends upon the use case application of the ANN. For example, in regression, a quadratic cost function is used:

$$C = \frac{1}{2n} \sum_x \left\| y(x) - a^L(x) \right\|^2 \tag{2.4}$$

Where n is the total number of training examples. The individual training examples are denoted by x. The corresponding desired output is *y=y(x). L* denotes the number of layers in the network and a^L=$a^L(x)$ is result obtained from the output layer of the the network.

In case of a binary classification, Cross-entropy loss function is used:

$$C = -[y(x) \log a^{\mathrm{L}}(x) + (1 - y(x)) \log \left(1 - a^{\mathrm{L}}(x)\right)] \tag{2.5}$$

The cost functions are designed in such a way that they are differentiable. A neural network *learns* by *changing* the weights and biases such that the cost function is *minimized*. The new values of weights and biases are calculated by *gradient descent*.

2.2.7 Gradient Descent

The goal of training a neural network is finding the values of weights and biases such that the cost function *C(w,b)* is minimized. The optimal values of weights and biases are found by a optimization method called *gradient descent*. To visualize the working of gradient descent, let consider the cost function *C* as a function of just two variables, *v1* and *v2*:

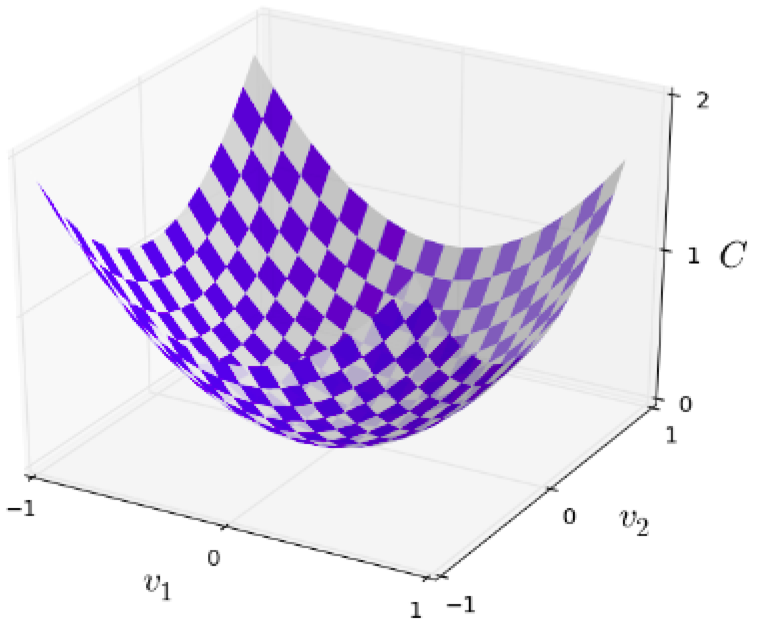


Figure 2.4: Cost function of 2 variables.

In order to minimize the value of cost function, we need to change the values of *v1* and *v2* such that we reach the minimum value of the function. From optimization theory[10], it is possible to calculate the relative change in the cost function C as a function of the slopes (technically, gradients) relative to the variables:

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2 \tag{2.6}$$

In order to understand the intuitive meaning behind the equation stated above, let us consider the value of the cost function represented by the green point in figure 2.5. From the green point, we need to move in the direction of the minimum (or visually, a *valley*) of the cost function. The direction is represented by the *gradient vector* $\bigtriangledown C$.

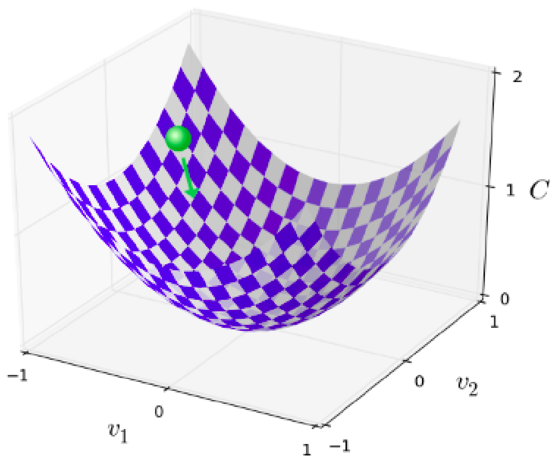


Figure 2.5: Visualizing gradient descent.

For a cost function with 2 variables, the *gradient vector* $\bigtriangledown C$ is:

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T \tag{2.7}$$

Calculation of the *gradient vector* gives us information about the direction of the minimum. Now that we know the direction of the minimum, we need to move in that direction. The movement is achieved by changing the values of *v1* and *v2*. The relationship between the change in a variable ($\bigtriangleup v$) and the gradient ($\bigtriangledown C$) is represented as:

$$\Delta v = -\eta \nabla C \tag{2.8}$$

Where η is a small, positive parameter known as the *learning rate*. η corresponds to the magnitude of the change in variable. If we select a *large* η, we move in a *larger step* direction of the gradient. If we select a *small* η, we move in a *smaller step* in direction of the gradient. The new value of v' is calculated from the existing value of v in the following way:

$$v \rightarrow v' = v - \eta \nabla C \tag{2.9}$$

After calculating the new values of *v1* and *v2*, the cost function is re-calculated and the entire process repeats iteratively. The process repeats till a *minimum threshold* of the cost function value is reached.

By extending a similar analogy to the ANN, we can find the required values of weights and biases in order to reduce the value of the cost function. In an ANN, The gradient vector $\bigtriangledown C$ will have the components $\partial C/\partial w_k$ and $\partial C/\partial b_l$. Writing out the gradient descent update rule in terms of weights and biases:

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \tag{2.10}$$

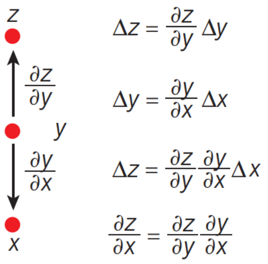
$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l} \tag{2.11}$$

The calculation of the gradient update rule for individual weights and biases is possible by *chain rule* from calculus.

2.2.8 Chain rule and Backpropagation

The main feature of a neural network is the ability to update its weights and biases such that particular inputs map to particular outputs. In backpropagation, the gradient descent from outer layers is passed through the inner layers using *chain rule* of calculus. To understand backpropagation, we need to understand the chain rule of calculus.

If Z is an output from the function Y, and Y is an output from the function X. Then the chain rule states that $\partial Z/\partial x$ is a product of $\partial Z/\partial y$ and $\partial y/\partial x$.



z
y
x
$\frac{\partial z}{\partial y}$
$\frac{\partial y}{\partial x}$
$\Delta z = \frac{\partial z}{\partial y}\Delta y$
$\Delta y = \frac{\partial y}{\partial x}\Delta x$
$\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x$
$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$

Figure 2.6: Back-propagation is an extension of the chain-rule from derivatives(ref:[28]).

In an ANN, the outputs of the outer layer are a function of the inner layers. Hence, the gradient of the cost function is *backpropagated* to the inner layers by the chain rule. Therefore, it is possible to calculate the partial derivatives of the cost function with-respect-to the individual weights ($\partial C/\partial w$) and individual biases ($\partial C/\partial b$).

$$w_k \rightarrow w_k' = w_k - \eta \frac{\partial C}{\partial w_k} \tag{2.12}$$

$$b_l \rightarrow b_l' = b_l - \eta \frac{\partial C}{\partial b_l} \tag{2.13}$$

2.2.9 Variations of ANNs

There are many variations of the ANN architecture being currently used in different domains. The 2-dimensional CNN model is used in image/ video classification[28]. The recurrent neural network (RNN) and 1-d CNN is used in time series analysis[56]. Self-organising maps (SOMs) are used to represent the similarity between data points[22] visually. Though there are differences in the architectures of the various ANN models, the underlying mechanics of *cost function*, *gradient descent* and *backpropagation* are similar. As the proposed architecture focuses solely on the 2-dimensional CNN variant of the ANN, the authors give a brief background on the 2-dimensional CNNs in the next section.

2.3 CNNs and the visual cortex

CNNs are a variation of the ANNs. CNNs take inspiration from the biological structure of the visual cortex[18]. The researchers in [18] observed the presence of two kinds of cells in the visual cortex: simple cells, which respond at a very local and particular spatial location, and complex cells, which pool inputs from multiple simple cells. The mechanical output of these two cells, the simple cells and the complex cells, forms the basis of convolutional neural networks.

2.3.1 Structure and working of a vanilla CNN variant

To understand how a CNN architecture automatically detects features from a given data, we need to understand the working of a CNN. There are 4 major operations in any CNN architecture:

1. Convolution

2. Activation function

3. Pooling/ sampling

4. Classification (Fully connected layer)

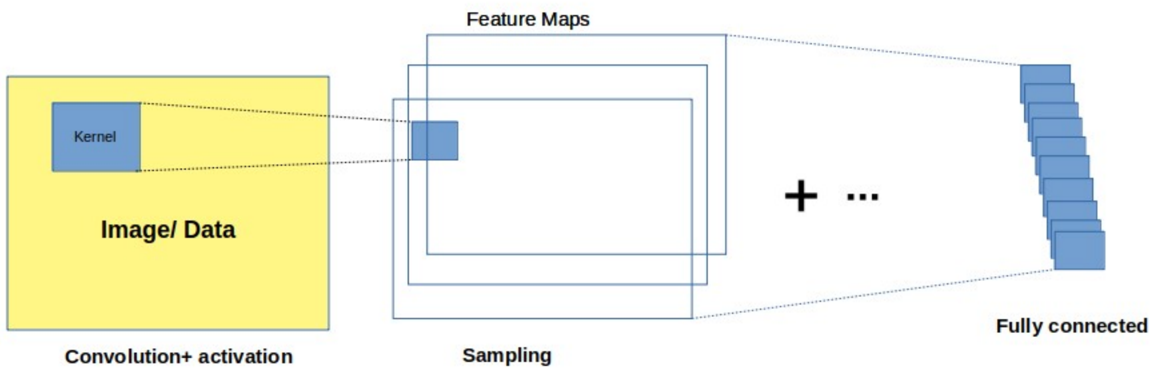


Figure 2.7: Overall structure of a vanilla CNN.

Convolution

For image classification, the input data to a CNN is an image. In convolution, a small matrix of numbers (called *kernel* or *filter*) is passed over the image/ data and transformed to generate *feature maps*. The feature map values are calculated according to the following formula:

$$G[m,n] = (f*h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k] \tag{2.14}$$

Where the input image is denoted by *f* and the kernel by *h*. The indexes of rows and columns of the result matrix are marked with *m* and *n* respectively.

The process of convolution is visually represented in figure 2.8. After placing our filter over selected pixels, we take each value from the kernel and multiply them in pairs with corresponding values from the image. Finally, we sum up everything and put the result in the right place in the output feature map.

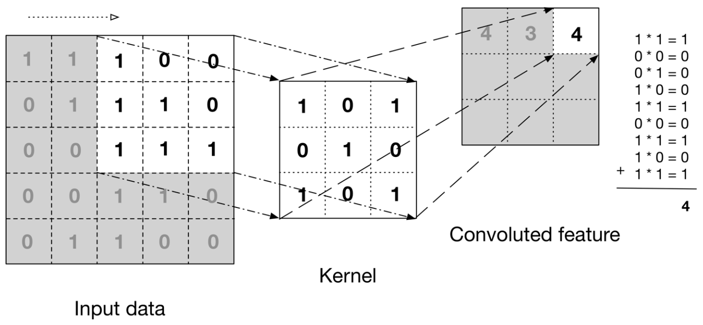


Figure 2.8: Visual representation of convolution.

Convolution in the edges of the image

In the example figure 2.8, when we perform convolution over the 5x5 image with a 3x3 kernel, we get a 3x3 feature map. The reason for a 3x3 feature map is because of the 9 unique positions where we can place our filter inside the image. In convolution, the impact of the pixels in the corner is much smaller than the centre of the image. To prevent the loss of information from the edges of the image, the input image is padded with an extra border. Usually, we fill in the padding with zeroes. Depending on whether we use padding or not, we are dealing with two types of convolution: *valid* and *same*. In *valid*, we use the original image. In *same*, we use the padded border around it, so that the images at the input and output are the same size.

Strides and pooling

Stride of a kernel refers to the number of pixels used to traverse in convolution. If *Stride = 1*, then the kernel is moved over one pixel to calculate the feature maps. *Pooling* refers to the downsampling of feature maps using various functions. Many different functions are used to downsample the feature maps, such as, *Max*, *Min* and *Average*. To illustrate pooling, the example to a *max-pooling* with *stride = 2* is shown below:

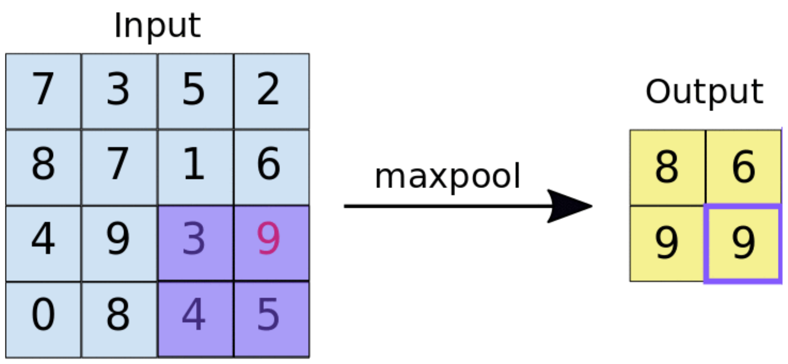


Figure 2.9: Visual representation of *maxpooling* with *stride* = *2*.

Activation functions

The equivalent of the firing in biological neurons is the presence of activation functions in the CNNs. Based upon the input value, the magnitude of the output is decided. Some of the common activation functions are illustrated below:

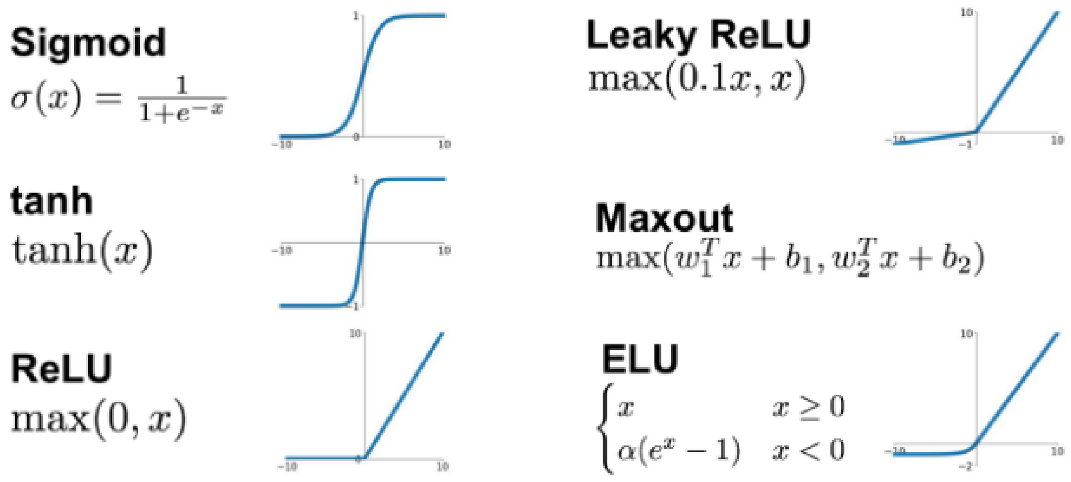


Figure 2.10: Some of the commonly used activation functions.

Fully connected layers

Generally, a combination of *convolution* + *activation* + *pooling* is repeated multiple times before the final layer(s) of *fully connected* (FC) neurons. Multiple convolutional layers (in series with activation functions and pooling layers) have been linked to identifying complex non-linear features from the images[28]. The FC neurons are similar to a vanilla ANN structure where all neurons from the penultimate layer are connected to the final layer. If the task at hand is classification, then the number of neurons of the outermost FC layer equals the number of classes to be predicted. If a single neuron is present in the outermost FC layer, then the task is to calculate continuous values. For binary classification, a sigmoid activation function is utilised. For multi-class classification, a softmax activation function is utilised in the FC layer.

Sigmoid and softmax activation function

The softmax function is a generalisation of sigmoid function for multi-class classification. Hence, for multi-class classification, a softmax activation function is preferred over the sigmoid function.

For a multi-class classification with K classes, the predicted probabilities using the softmax function are as follows:

$$\Pr(Y_i = k) = \frac{e^{\beta_k \mathbf{X}_i}}{\sum_{0 \le c \le K} e^{\beta_c \mathbf{X}_i}} \tag{2.15}$$

For a binary classification we replace β= -(β_0 -β_1). After substitution, we obtain the same probabilities as in the sigmoid activation function.

$$\begin{aligned}\Pr\left(Y_i = 0\right) &= \frac{e^{\beta_0 \cdot \mathbf{X}_i}}{\sum_{0 \le c \le K} e^{\beta_c \cdot \mathbf{X}_i}} = \frac{e^{\beta_0 \cdot \mathbf{X}_i}}{e^{\beta_0 \cdot \mathbf{X}_i} + e^{\beta_1 \cdot \mathbf{X}_i}} = \frac{e^{(\beta_0 - \beta_1) \cdot \mathbf{X}_i}}{e^{(\beta_0 - \beta_1) \cdot \mathbf{X}_i + 1}} = \frac{e^{-\beta \cdot X_i}}{1 + e^{-\beta \cdot X_i}} \\ \Pr\left(Y_i = 1\right) &= \frac{e^{\beta_1 \cdot \mathbf{X}_i}}{\sum_{0 \le c \le K} e^{\beta_c \cdot \mathbf{X}_i}} = \frac{e^{\beta_1 \cdot X_i}}{e^{\beta_0 \cdot \mathbf{X}_i} + e^{\beta_1 \cdot \mathbf{X}_i}} = \frac{1}{e^{(\beta_0 - \beta_1) \cdot \mathbf{X}_i + 1}} = \frac{1}{1 + e^{-\beta \cdot \mathbf{X}_i}}\end{aligned} \tag{2.16}$$

2.3.2 Automatic feature extraction in CNNs

In chapter 2.2, the authors outlined the architecture of a vanilla ANN architecture. The weights and biases of the ANN are updated using gradient descent and backpropagation such that the error between predictions (cost function) is reduced. Hence, using the weights and biases, the ANN builds non-linear mappings between the inputs and outputs. Even though the architecture of a CNN model is different from ANN, the weights and biases are updated using the same concepts of gradient descent and backpropagation.

A study by Zeiler et al. [58] states that the various layers of any CNN learn the essential features during their training process. In the case of image classification, the layers closer to the input learn basic features such as points, line. The layers deeper into the CNN architecture combine the simpler features from the previous layers to identify complex features such as the shapes of the forehead, nose, among others. Each layer of the CNN automatically detects important features and complex features are found in the deeper layers of the CNN. The process of automatic feature detection removes the latency in designing complex handcrafted feature engineering techniques and transfers it to the training process of the CNN model.

2.3.3 CNNs as image classifiers

The convolution layers, activation function, sampling and fully connected layers form the basic building blocks of any CNN architecture. By varying the different blocks and adjusting the number of neurons and layers, we can generate many different variants of the vanilla architecture. There are certain standard architectures such as LeNet[28], Imagenet[25] and VGG[48], to name a few. The architectures of LeNet, ImageNet and VGG have outperformed handcrafted feature engineering methods in image classification and object detection tasks. Hence, the strength of CNNs lies in automatically detecting features in the given data.

2.4 Recurrence Plots

In chapter 2.1.7, the authors had discussed the need to develop a PD identification system which not only has the capability to *automatically extract features*, but is also *robust to the presence of noise and outliers*. In the previous section, the authors concluded that the strength of CNNs lies in automatic feature engineering. To address the requirement of *raw data handling capability*, the authors give a brief background on recurrence plots (RPs). In the current section, the authors discuss the working of RPs, how RPs visually represent temporal data, and how RPs are robust to the presence of outliers in the data.

2.4.1 Introduction

Recurrence plots are tools used to visualise the temporal relationships between dynamical systems[15]. It is a technique of non-linear data analysis which is used to inspect temporal correlations in a univariate time series data visually. An RP is visualised as a square matrix, in which the matrix elements correspond to those times at which a state of a dynamical system recurs (columns and rows correspond then to a certain pair of times). In the case of time-series data from a sensor, the different states will correspond to the different amplitudes of the signal.

2.4.2 Constructing RPs

The algorithm for constructing a RP is given below:

$$R(i,j) = \begin{cases} 1 & \text{if} \quad \|x(i) - x(j) \leq \varepsilon\| \\ 0 & \text{elsewhere} \end{cases} \tag{2.17}$$

Where *i* and *j* are two time points taken into consideration. *x(i)* and *x(j)* represent the values of the signal at the time points *i* and *j*. If *x(i)* and *x(j)* are *similar*, then we mark a point at the corresponding (i, j) coordinates in the RP. The choice of the measure of similarity depends on the user, but euclidean distance is a preferred metric over manhattan and mahalanobis distances. The similarity is bound by a threshold (the ε in the eq: 2.17). Hence, two points in the state space are similar, if the distance between them is less than or equal to the ε. The *closeness* in similarity corresponds to a high *temporal correlation*. Hence, if a point exists at coordinates (i, j), a temporal correlation exists for the times i and j. The value of ε is empirically determined and is generally capped at 10% of the mean of the data[30].

2.4.3 Examples of RPs

Illustrative examples of RPs for two different functions are represented in the figures 2.11 and 2.12. From the figures 2.11 and 2.12, we can see that the RPs for periodic functions have distinct patterns (figure 2.11). There are no distinct patterns in the case of white noise (figure2.12).

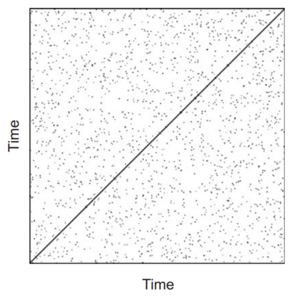


Figure 2.11: RP of white noise,(ref:[31])

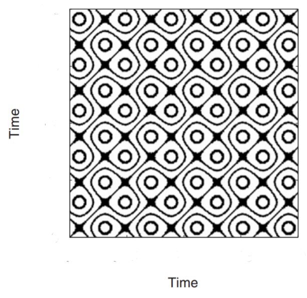


Figure 2.12: RP of equation cos(2π1000t + 0.5 sin(2π25t)),(ref:[31])

2.4.4 Robustness against outliers

If the signal is periodic, then RPs can negate the effect of outliers. To illustrate this, let us consider two sinusoidal signals which are similar except for the presence of an outlier between the time 50 and 55 (refer figures 2.13 and 2.14).

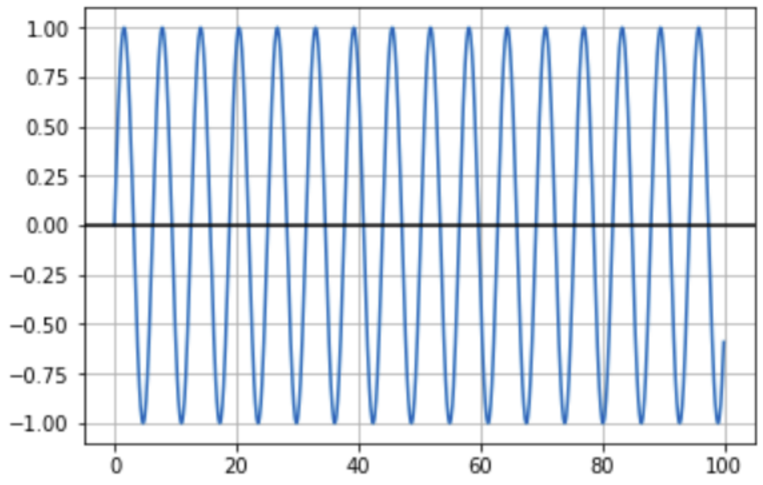


Figure 2.13: A sine function.

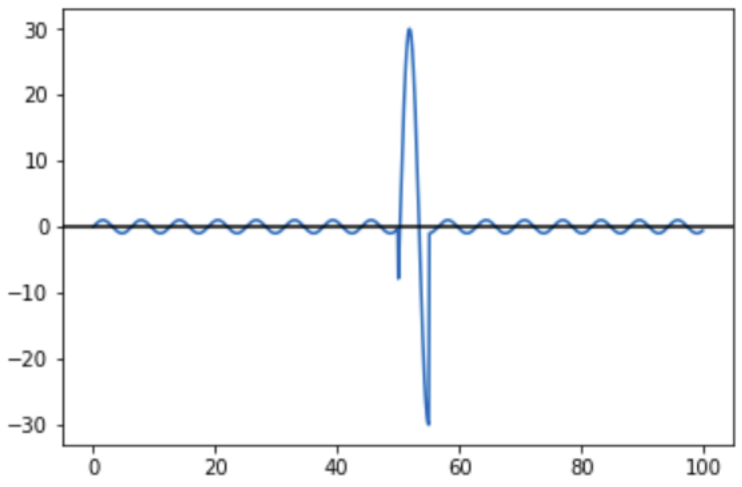


Figure 2.14: Sine function with outlier at t=(50,55).

We construct the RPs for the two sinusoidal signals with a granularity of 1000 data points. This means we will have 1000 time points on the x and y-axis of the RPs. From the RPs, we can observe that the periodicity of the waveform is maintained and circular structures present in the RPs represent it (refer figures 2.19 and 2.20). A dark line represents the outlier values present at time 50 through 55 (500 through 550, due to the granularity). This way, we observe that the presence of outliers does not affect the overall temporal representation of the RP.

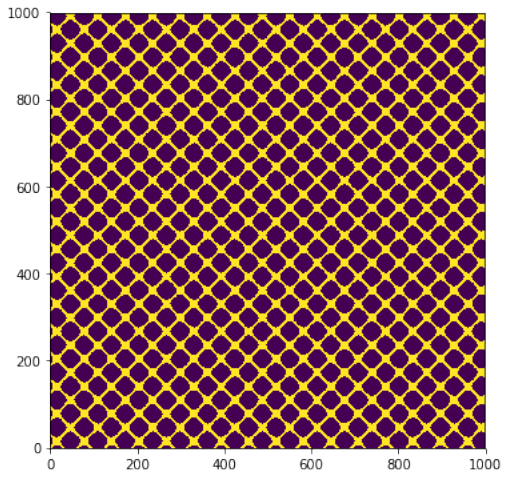


Figure 2.15: A sine signals.

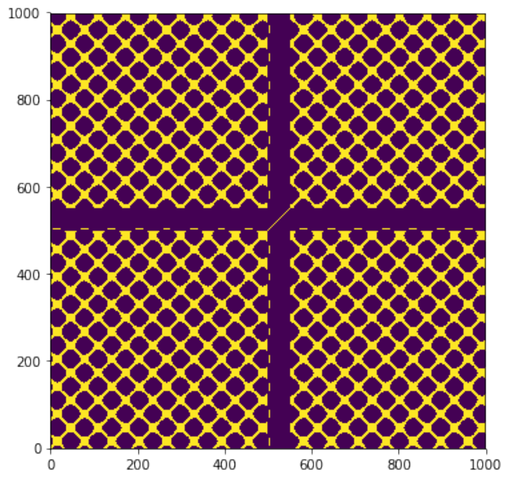


Figure 2.16: Sine signals with outlier at t = (50,55).

2.4.5 Influence of noise on RPs

Apart from the robustness against outliers, RPs can also negate the effect of noisy data. To illustrate this, let us consider two sinusoidal signals: one is a pure sinusoidal wave, and the other is a noisy sinusoidal wave (refer figures 2.17 and 2.18).

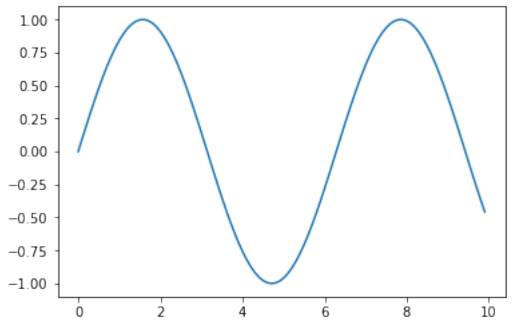


Figure 2.17: A pure sine function.

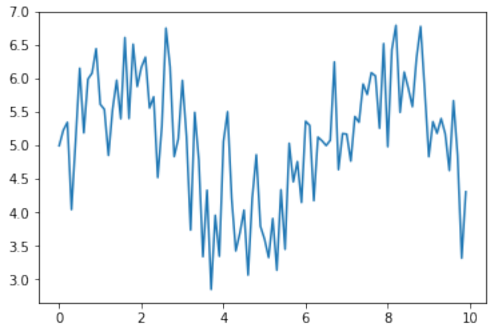


Figure 2.18: A noisy sine function.

We construct the RPs for the two sinusoidal functions with a granularity of 100 data points. From the RPs, we can observe that even though the RP for noisy signal contains elements of noisy data, the overall structure is maintained (refer figures 2.19 and 2.20). We observe that even though the noisy RP is not the same as pure sine wave RP, the noisy RP still contains the traces of a pure RP signal.

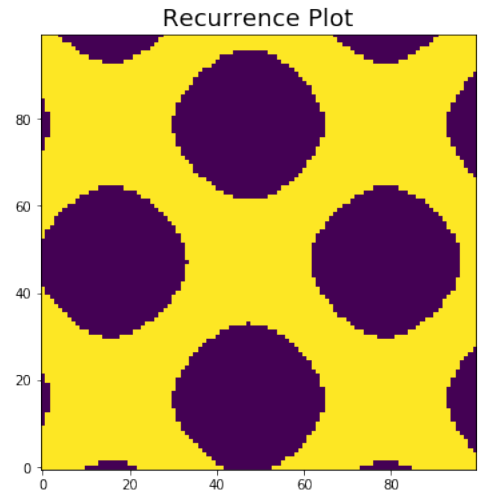


Figure 2.19: RP of pure sine function.

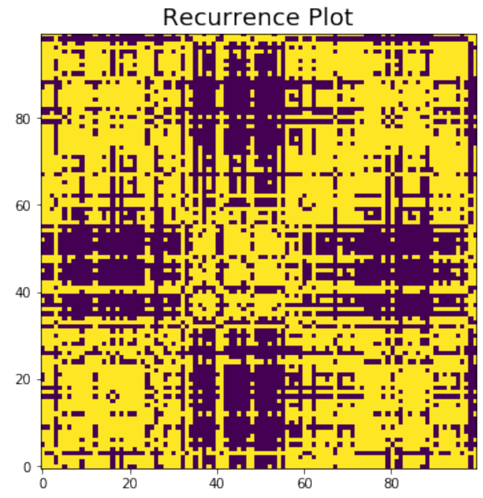


Figure 2.20: RP of noisy sine function contains traces of pure sine.

2.4.6 Visual features of temporal data

The previous sections give illustrations of how an RP representation of a periodic signal is robust against the presence of noise and outliers. Another strength of RPs is the visual representation of the signal behaviour. Marwan et al. mention 9 different RP patterns and how each pattern corresponds to a distinct signal behaviour[30]. For instance, the presence of *fading lines in the upper left/ lower right corners* of the RP corresponds to the signal containing a *trend or a drift*. Thereby, 9 such distinct RP patterns can be visually observed to infer 9 different characteristics of the signal. Hence, RPs can *translate signal behaviour into visually observable patterns*. The authors exploit the ability of RPs to visually store temporal information by combining the architecture of RPs and CNNs; this is explained in chapter 4.

2.4.7 Drawbacks

There has been substantial amount of research work in RP applications[26][16]. Researchers have also applied feature extraction processes on RPs, called recurrence quantification analysis (RQA)[57][32][9]. Though the strength of RPs lies in the robustness of the visual representation of periodic signals against outliers, it has two major drawbacks. One of the drawbacks is handling missing values. If data is missing, for example, loss of signal data, then data pre-processing needs to be performed. Another major challenge in RP generation is the calculation of

ε in the equation 2.17. The empirical calculation of ε proves to be a hindrance towards the goal of creating a PD identification technique with *raw data handling capability*.

Chapter 3

Literature review and related work

3.1 Literature review of computer-aided PD identification

Identification of PD is carried out by identifying the various motor and non-motor related symptoms of the disease. For computer-aided detection, quantifying and capturing the motor-related symptoms is easier than non-motor symptoms. Depending on the nature of the symptom, different technologies are utilised to capture specific data. For example, tremor in the voice, which corresponds to dyskinesia, is captured using microphones. FOG, which corresponds to akinesia, is captured by either motion sensors or by analysing the image/video data of the patients. The tremor in hands, which also corresponds to dyskinesia, is captured by motion sensors on the arms and wrists .

3.1.1 Selection criteria for inclusion of literature

A consolidated overview of state-of-the-art research work in computer-aided PD identification presented in table 3.1. It should be noted that the research work in table 3.1 is not an exhaustive list. The research work present in table 3.1 satisfies at least one of the following criteria:

1. **Research contribution in the form of a new dataset**: The research work provides a dataset which can be utilised by other researchers.
2. **Research contribution in the form of a "new" algorithm**: The research work provides comparable results in PD identification by utilizing a *different or/ new* algorithm.
3. **Research contribution in the form of comparable performance metrics**: The performance metrics of the research work is close to/ better than the state-of-the-art results (at the time of publication).

3.1.2 Observations from the literature review

1. **On body sensors are popular tools to capture data.**

Accelerometers and gyroscopes are most commonly used sensors to capture data for motor-related symptoms in PD. Hoff et al. [17], Keijsers et al.[23], Patel et al.[36], Cancela et al.[13], Tsipouras et al.[50], Zwartjes et al.[59], Roy et al.[44], Cook et al.[14], Bachlin et al.[7], Mazilu et al.[33], Jane et al.[20], Albert et al.[4] and Tripoliti et al.[49] utilize accelerometers to obtain data about the movement of the subject.

Little et al.[29] makes use of MDVP, an acoustic software tool for capturing acoustic features of speech in PD patients. Wahid et al.[53] capture video of PD subjects walking on force plates for spatio-temporal evaluation of gait. Cook et al.[14] also utilises video and image sensors as part of "smart monitoring" of the subjects. Zwartjes et al.[59] utilise gyroscopes to capture data. Mazilu et al.[33] and Albert et al.[4] utilize accelerometers present in smartphones, to capture data. Apart from utilising accelerometer data, Roy et al.[44] use surface electromyography (EMG) also as part of their research work.

2. **Pre-processing involves removing higher frequency components.**

As part of the pre-processing, higher-order components of the signals are removed to reduce noise in the data. Hoff et al. [17], Keijsers et al. [23], Bachlin et al.[7] considers only the frequency band of (1Hz-8Hz). Patel et al.[36] considers two separate bands of frequencies: <1Hz and (3Hz-8Hz). Mazilu et al.[33] too considers two separate bands of frequencies: (0.5Hz-3Hz) and (3Hz-8Hz). Wahid et al.[53] utilizes a 4Hz low pass filter. Tripoliti et al.[49] utilizes components in the band (2Hz-10Hz). Zwartjes et al.[59] utilised different frequency bands for different analysis, the upper limit is capped at 12Hz. Roy et al.[44] used a 5Hz low pass filter.

3. **Complex handcrafted feature extraction methods.**

As the PD identification systems are designed to identify a particular set of symptoms, complex handcrafted feature engineering processes are a part of the system. A major portion of the research work involves a thorough study of the data. By a thorough understanding of the data, handcrafted feature engineering methods are fine-tuned to identify the complex patterns in the data.

Hoff et al. [17], Keijsers et al.[23], Patel et al.[36], Roy et al.[44], Mazilu et al.[33] and Bachlin et al.[7] calculate statistical and frequency domain features for different frequency bands.

Hoff et al. calculates correlation of 2 different frequency bands : (1Hz-4Hz) and (4Hz-8Hz) with 4 activities of daily living (ADL). Keijsers et al.[23], Patel et al.[36], Cancela et al.[13], Tsipouras et al.[50], Zwartjes et al.[59], Roy et al.[44], Mazilu et al.[33], Albert et al.[4], Tripoliti et al.[49] and Cook et al.[14] utilize statistical and frequency features. The statistical features include the mean, median, standard deviation, cross-correlation to name a few. The frequency features include power spectral density, Fourier transform coefficients, to name a few.

As mentioned before, Little et al.[29] utilises MDVP, an acoustic software tool, generating features as MDVP(jitter), recurrence period density entropy (RPDE), pitch period entropy, to name a few. The extracted features are used as an input to the support vector machine (SVM), which reported an overall accuracy of 91.4%. Apart from Little et al., Patel et al., Cancela et al., Albert et al., Wahid et al. and Cook et al. have utilised SVM for classifying the extracted features.

Wahid et al. calculated features regarding the stride and gait such as stride time, stride length, to name a few. After calculating the accuracies of SVM, random forests (RF), naive bayes(NB) and k-nearest neighbours(kNN), they report the highest accuracy of 92.6% using random forests.

Jane et al.[20] use a variant of ANNs called Q-Backpropagated time-delay neural networks (QBTDNN) in which the mechanism of backpropagation is different from a vanilla neural network. Jane et al. do not perform any handcrafted feature engineering or outlier treatment. The QBTDNN architecture reports the highest classification accuracy of 92.19%.

Table 3.1: Consolidated list of research work related to PD identification.

Study	**Year**	**Reference**	**Sample**	**Symptoms**	**Sensors**	**Feature engineering/ Outlier treatment (OT) ?**	**Algorithm**	**Results**
Hoff et al.	2001	[17]	23 PD	dyskinesia	accelerometers	OT + frequency features	linear regression	correlation coefficient of 0.83, 0.87, 0.82 while sitting, counting, and spelling.
Keijsers et al.	2003	[23]	13 PD	dyskinesia	accelerometers	OT + statistical features + frequency features	ANN	classification accuracy of 93.7%, 99.7%, and 97.0% in the arm, trunk, and leg.
Little et al.	2009	[29]	23 PD,8 HC	dyskinesia	microphone	OT + statistical features + frequency features	SVM	overall classification accuracy of 91.4%
Patel et al.	2009	[36]	12 PD	bradykinesia + dyskinesia	accelerometers	OT + statistical features + frequency features	SVM	percent estimation error of 2.8% for tremor, 1.7% for bradykinesia, and 1.2% for dyskinesia

Cancela et al.	2010	[13]	20 PD	bradykinesia	accelerometers + gyroscopes	OT + statistical features + frequency features	KNN, ANN, DT, SVM	highest accuracy of 86.48% for SVM
Tsipouras et al.	2010	[50]	7 PD,3 HC	dyskinesia	accelerometers + gyroscopes	OT + frequency features	NB, DT, KNN, RF	highest accuracy of 93.73 for RF
Zwartjes et al.	2010	[59]	6 PD,7 HC	bradykinesia + dyskinesia	accelerometers + gyroscopes	OT + frequency features	DT	classification accuracy of 99.3% for activity classifier
Roy et al.	2011	[44]	19 PD,4 HC	dyskinesia	accelerometers + gyroscopes + EMG	OT + frequency features	ANN	mean sensitivity and specificity of 90.5% and 93.2%
Mazilu et al.	2012	[33]	10 PD	akinesia	accelerometers + gyroscopes	OT + frequency features	RF, NB, DT, MLP	highest sensitivity and specificity of 97.76% and 99.75% was recorded for RF
Albert at al.	2012	[4]	8 PD,18 HC	bradykinesia + dyskinesia	acc	OT + frequency features	SVM	classification accuracy of 87% for SVM
Jane et al.	2016	[20]	93 PD,73 HC	bradykinesia + akinesia	trans - ducers	OT	ANN	highest classification accuracy of 92.19%

Tripoliti et al.	2013	[49]	11 PD,5 HC	akinesia	accelerometers + gyroscopes	OT + frequency features	RF, NB, DT	highest classification accuracy of 96.11% for RF
Wahid et al.	2015	[53]	23 PD,26 HC	bradykinesia + akinesia	cameras + transducers	OT + statistical features + frequency features	KNN, SVM, NB	highest accuracy of 92.6% was observed in RF
Cook et al.	2015	[14]	50 PD, 68 HC	bradykinesia + dyskinesia	accelerometers + gyroscopes + transducers	OT + statistical features + frequency features	DT, RF, SVM, NB	highest accuracy of 85% was observed in DT with AdaBoost
Bachlin et al.	2010	[7]	10 PD	akikinesia	accelerometers	OT + frequency features	Energy Thresholding	mean sensitivity and specificity of 73.1% and 81.6%

It is evident from table 3.1 that most of the research works utilise specific handcrafted feature engineering processes. Apart from the presence of handcrafted feature engineering processes, most of the research works also involve the use of filters to remove noise and outliers from the data.

3.2 Related literature on computer-aided akinesia detection

In chapter 3, the authors have outlined the reasons for selection data from Bachlin et al.[7] for evaluation of the proposed architecture. As the dataset in Bachlin et al. corresponds to akinesia detection, the authors have summarised relevant literature for akinesia detection in table 3.2. It should be noted that the list of research work in table 3.2 is not exhaustive, but follows filtering criteria similar to the one used in table 3.1. The research work in table 3.2 satisfies at least one of the following criteria:

1. **Different/New algorithm for identification**: in comparison to the other research works at the time of publishing, the research work utilizes a *different/ new* identification technique .
2. **Comparable performance metrics values**: The performance metrics of the research work are close to/ better than the state-of-the-art results (at the time of publication)

The result of the filtering, as mentioned earlier, is populated in table 3.2.

Table 3.2: Consolidated list of research work in automatic detection of Akinesia/FOG

Study	**Year**	**Reference**	**Sample**	**Sensors**	**Feature engineering/Outlier treatment (OT)?**	**algorithm?**	**Results**
Bachlin et al.	2010	[7]	10 PD	accelerometers	OT + frequency features	Energy Thresholding	mean sensitivity and specificity of 73.1% and 81.6%
Ahlrichs et al.	2016	[3]	20 PD	accelerometers + gyroscopes	OT + frequency features	SVM	classification accuracy of 95.4%
Kim et al.	2015	[24]	15 PD	accelerometers	OT + statistical features + frequency features	RF	highest classification sensitivity and specifiaccelerometerscity of 86% and 91.7%
Alsheikh et al.	2016	[6]	10 PD	accelerometers	frequency features	DBN	highest classification accuracy of 97.85%
Ravi et al.	2016	[40]	10 PD	accelerometers	frequency features	CNN	sensitivity and -specificity of 91.5% and 97.7%
Rezva-nian et al.	2016	[41]	10 PD	accelerometers	OT + frequency features	Energy Thresholding	highest sensitivity and specificity of 84.9% and 81% for ankle/ shank sensors
Camps et al.	2018	[12]	21 PD	accelerometers + gyroscopes	OT + frequency features	1d-CNN, RF, SVM	accuracy, sensitivity and specificity for validation set is 87.9%, 92.6% and 88.7%

Wu et al.	2016	[55]	29 PD, 29 HC	transducers	OT + frequency features	SVM. GLRA	highest classification accuracy of 84.48% for SVM
Saad et al.	2013	[45]	10 PD	accelerometers	OT + frequency features	BBN	average classifcation accuracy of 74.31%
Pham et al.	2017	[38]	10 PD	accelerometers	OT + frequency features	Spectral coherence	sensitivity and specificity of 66.25% and 95.38%
Sansegundo at al.	2019	[46]	10 PD	accelerometers	OT + frequency features	CNN + HMM	CNN model scored sensitivity and -specificity of 99.5% and 99.99%
Xia et al.	2019	[56]	10 PD	accelerometers	OT	1d-CNN	Average sensitivity and specificity for "cross-patient" reported as 69.29% and 90.60%
Afonso et al.	2019	[1]	4 PD,11 HC	accelerometers + gyroscopes	None	RP + CNN	mean sensitivity and specificity of 92.25% and 93.65%

3.2.1 Observations from related literature

1. **Filtering data to remove noise/ outliers.**

Though the PD identification techniques involve handcrafted feature engineering techniques, filtering the data to remove noise and outliers is one of the most critical steps as part of the data pre-processing. Ahlrichs et al. and Kim et al. use two different filters of (0.5Hz-3Hz) and (3Hz-8Hz). Rezvanian et al. uses 10Hz low pass filters. Similarly, Camps and San-Segundo et al. utilise a 20Hz filter.

2. **Complex handcrafted feature engineering.**

Generally, handcrafted feature engineering techniques are designed after a thorough understanding of the data. Many times, outlier treatment and feature engineering complement each other, for example: in Kim et al., Bachlin et al. and Ahlrichs et al. calculate statistical and frequency domain features for (0.5Hz-3Hz) and (3Hz-8Hz) frequency bands separately. Xia et al. use 4-σ rule to identify outliers and replace them with the corresponding median values.

ANN and their variants are also used for classification, but the input data is transformed into the frequency domain. For instance, Camps et al., Ravi et al., San-Segundo et al. and Rezvanian et al. utilise CNN architectures, but the raw data is transformed into frequency domain as part of the feature extraction process. Rezvanian et al. utilises features described by Bachlin et al. in conjunction with a newly proposed method which involved continuous fourier transform (CWT) of the data. Ravi et al. and Camps et al. use spectrogram representation of the data. Camps et al. uses 1-dimensional CNN, whereas Ravi et al. use a 2-dimensional CNN architecture for classification.

Xia et al. uses a 1-dimensional CNN architecture without any handcrafted features. The researchers of Xia et al. evaluated the *learning rate*, *momentum*, *signal window size*and *kernel size* of the 1-d CNN architecture.

Afonso et al. is the only paper, in regards to akinesia detection, which does not employ either any handcrafted feature engineering or specific outlier treatment on the data. A "smartpen" is used to extract data, which is then utilised to construct RPs. The RPs are constructed using varying window sizes (0.96 seconds and 1.92 seconds, respectively). The generated RPs are used an input to train standard CNN architectures: LeNet, CIFAR-10 and ImageNet.

Chapter 4

Dataset

4.1 Introduction

The dataset is a contribution of the research work of Bachlin et al. [7]. The authors have outlined the reasons for selecting the dataset in the next section. After outlining the reasons for selecting the dataset, the authors briefly explain how the dataset was collected and what each data-point represents.

4.2 Reasons for selecting Bachlin et al. dataset

To evaluate the performance of the RPCNN architecture, the authors decided to select data from Bachlin et al. [7]. The authors have outlined the following reasons for selecting data from Bachlin et al.:

1. **Publicly available**: The data is publicly available at the UCI machine learning repository.

2. **Large amount of available research**: As the dataset is relatively old, the authors have access to a large number of research works which can be used as benchmarks against the current architecture.

3. **Standarad dataset for akinesia detection**: Due to the dataset being relatively old and being publicly available, the research community considers Bachlin et al. dataset as the standard dataset for akinesia detection.

4.3 Information about the subjects in the study

For the collection of data, ten patients with PD took part in the study. They consisted of seven males and three females. The age distribution of the patients is 66.4$\pm$ 4.8 years. The details about the age and duration of the disease are given in table 4.1. Motor performance of PD patients is associated with significant variability, and this is reflected in the data shown below.

Table 4.1: Age and duration of disease for each subject

Subject	**Gender**	**Age**	**Disease duration (in years)**
1	M	66	3
2	M	67	2
3	M	59	2.5
4	M	62	3
5	M	75	2
6	F	63	2
7	M	66	2.5
8	F	68	4
9	M	73	2
10	F	65	3
Mean±std		66.4±4.8	13.7±9.67

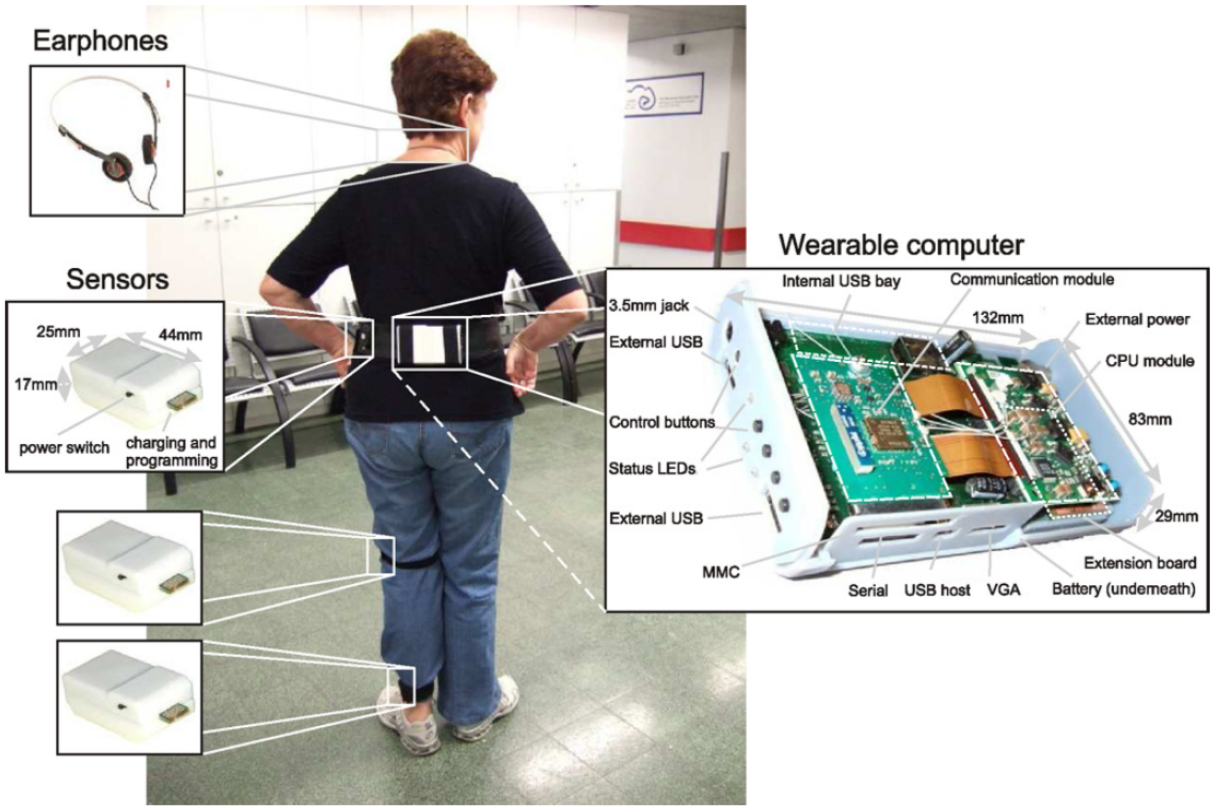


Figure 4.1: Three sensors were located at the ankle, knee and waist(ref [7])

4.4 How the data was collected

Bachlin et al. developed a wearable assistant for the detection of FOG events. The wearable assistant is a customised microprocessor mounted on a printed circuit board with accelerometers and rechargeable lithium-ion batteries. It has a universal serial bus (USB) and Bluetooth interfacing. The acquired data is transmitted over a wireless Bluetooth link (64 Hz) for online data processing.

The subjects are asked to perform three basic tasks(refer figure 4.2):

1. Walking back and forth in a straight line across the laboratory.

2. Random walking in an open space where an experimenter would give the patients commands to turn or stop, at random.

3. Walking simulating ADL, in which the patients were asked to enter and leave doors, getting something to drink and returning with a cup of water.

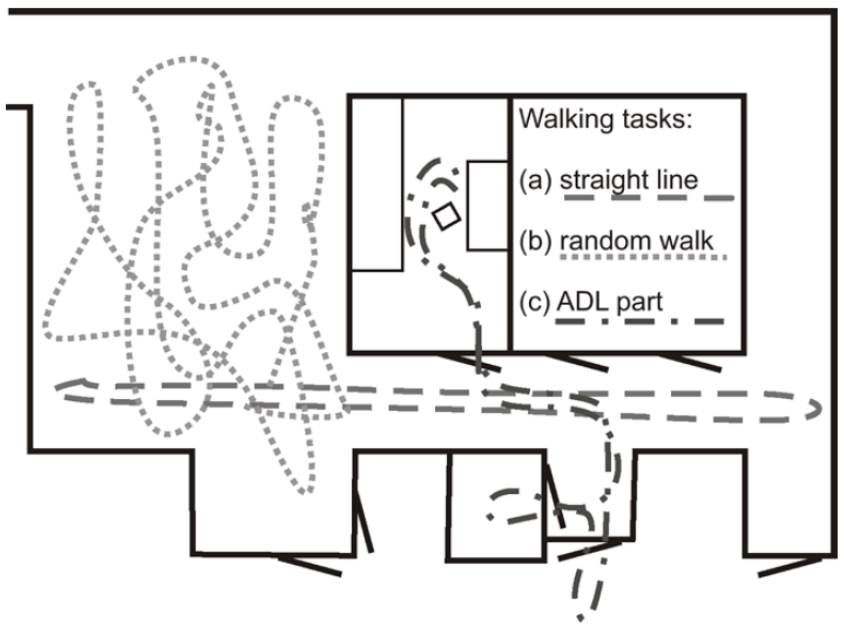


Figure 4.2: Walking and activity of daily living(ADL) tasks given to the subjects(ref [7]).

The accelerometers were attached to the knee, ankle and the waist/trunk of the subject(refer figure 4.1). The data was sampled at 64Hz and sent to the data processing unit via a Bluetooth connection. Hence, the raw data consists of data-points spaced at 15-millisecond intervals. Each of the accelerometers gives three-dimensional data. Hence each instance has nine columns of data. The tenth column gives us information about the "status" of the subject. The three categories or labels for the "status" at any point of time:

1. **Normal (non-FOG)**: The patient does not experience any FOG event during this time. This is encoded by the value "1" in the dataset

2. **FOG**: The patient experiences FOG, the patients were videotaped, and the data was analysed by physiotherapists to verify the exact start and stop of FOG events. In the data, this is encoded by "status" = "2".

3. **Not part of experiment**: The data is not part of the experiment. To segregate non-experimental data, the researchers labelled it as "non-experiment". These values are encoded as "0" in the dataset.

The table 4.2 gives information regarding the data extracted via the accelerometers. Each data point corresponds to the 15-millisecond timestamp(64Hz). We can also see in the table 4.2 that many of the subjects were asked to perform the walking and ADL tasks multiple times. Subjects 4 and 10 did not experience any FOG even though they have had the disease for three years until the date of the experiment.

Table 4.2: FOG and non-FOG related event duration for each subject.

Subject	No. of rounds	Non-FOG data points	FOG data points
1	2	114909	6694
2	2	78953	11609
3	3	110304	18340
4	1	132482	0
5	2	103396	30370
6	2	118952	8413
7	2	97829	5213
8	1	36425	12859
9	1	94078	17287
10	1	142722	0

4.5 Standard deviation of the raw and filtered data

As mentioned in chapter 5, the authors study the behaviour of the proposed architecture on both raw and filtered data. By using both noisy and filtered data, the authors aim to study the effect of outliers on akinesia identification. To generate the filtered data, the authors utilised the same process as mentioned in Bachlin et al. The researchers in Bachlin et al. used a filter to capture the components with frequencies between (0 to 8) Hz. After filtering the data to only include the low-frequency components, there is a change in the standard deviation of the raw data. The *standard deviation of the raw data* and the corresponding *percentage change in value after filtering* is shown in the tables 4.4 through 4.5.

Table 4.3: standard deviation and % change for filtered IMU measurements taken from the ankle.

Subject	**Axis_1**		**Axis_2**		**Axis_3**	
	Std	**% change**	**Std**	**% change**	**Std**	**% change**
1	586	201	357	167	313	415
2	649	156	439	283	312	374
3	502	234	343	182	284	282
4	302	189	210	317	234	345
5	494	273	358	333	294	481
6	588	312	379	152	351	267
7	457	298	271	328	255	319
8	244	450	120	295	90	231
9	409	346	278	357	237	284
10	702	341	546	263	478	349

Table 4.4: standard deviation and % change for filtered IMU measurements taken from the thigh.

Subject	**Axis_1**		**Axis_2**		**Axis_3**	
	Std	**% change**	**Std**	**% change**	**Std**	**% change**
1	520	251	344	228	222	382
2	616	346	335	158	243	284
3	522	184	423	337	217	326
4	449	251	387	361	141	376
5	450	367	236	382	203	428
6	596	286	470	284	450	214
7	386	321	214	359	132	270
8	198	239	85	239	83	213
9	353	381	232	437	273	371
10	416	349	546	256	282	350

Table 4.5: standard deviation and % change for filtered IMU measurements taken from the waist.

Subject	**Axis_1**		**Axis_2**		**Axis_3**	
	Std	**% change**	**Std**	**% change**	**Std**	**% change**
1	199	201	181	167	185	384
2	174	156	188	283	174	271
3	240	234	125	182	251	330
4	256	189	100	317	193	439
5	189	273	167	333	295	484
6	201	312	131	152	115	361
7	151	298	102	328	142	210
8	97	450	58	295	90	430
9	152	346	103	357	135	376
10	145	341	427	263	267	210

From the tables above, we can see that filtering out high-frequency noise has changed the standard deviation in the data. In the next chapter, the authors explain the approach of removing the dependency on both *complicated handcrafted feature engineering techniques* and *explicit outlier treatment*. Apart from explaining the theoretical basis of the proposed architecture, the authors also explain the various parameters and evaluation strategies.

Chapter 5

Approach

5.1 Introduction

As outlined in chapter 2, many of the current PD identification systems involve the use of either complicated handcrafted feature engineering or specific outlier removal techniques or both. The process of handcrafted feature engineering generally involves a thorough data analysis, followed by a suitable transformation of the data. The classification step only occurs after the analysis and transformation of the data is complete. This makes the process of analyzing and implementing any handcrafted feature engineering technique, complicated and arduous. Similar to handcrafted feature engineering, specific outlier removal techniques also involve a thorough understanding of the data to discern outliers and noise.

In chapter 2, the authors stress the importance of a timely diagnosis of PD.

Hence, there is a need to develop *faster* PD identification systems by reducing the dependency on both *complicated handcrafted feature engineering techniques* and *explicit outlier treatment*. To reach the goal of removing the dependency on both *complicated handcrafted feature engineering techniques* and *explicit outlier treatment*, the authors propose an architecture composed of RPs and CNNs.

5.2 Strengths of RPs

RPs are tools used to visualize the temporal correlation in univariate data. As mentioned in chapter 2, the characteristic feature of RPs is to represent the temporal behaviour of the data visually. As mentioned in chapter 2, Marwan et al. summarize 9 different RP patterns with 9 different behaviours of the data[30]. The authors have also shown that RP representation of periodic signals is robust against the presence of noise and outliers.

5.3 Strengths of CNNs

CNNs are a variant of ANNs. As mentioned in chapter 2, one of the main characteristics of CNNs is to detect patterns in the given data automatically. In image classification tasks, different variations of vanilla CNN models have proved their mettle against the handcrafted feature engineering approaches.

5.4 RPs and CNNs complement each other

As mentioned in chapter 2, Marwan et al. summarize 9 different patterns in RPs corresponding to 9 different data behaviours. If the human eye can associate 9 different features with 9 different data behaviours, then we can safely assume that a CNN model can uncover more hidden patterns. This is because the CNN architectures excel in finding complex non-linear relationships between the different data points. The ability of CNNs to uncover complex patterns complements the RP's ability to represent temporal data visually. Hence, the authors utilize the strengths of RPs and CNNs to develop a PD identification system which is not only *robust to the presence of outliers* but also performs *automatic feature engineering*.

In a recently published paper, Afonso et al. utilize RPs in conjunction with CNNs to identify bradykinesia in PD patients[37]. Similar to our approach, Afonso et al. use raw data directly without any prior handcrafted feature engineering. Though the proposed approach of the research work is similar to Afonso et al., there are differences in the implementation *architecture* and *evaluation strategy*.

The architecture of Afonso et al. involves the use of 3 different architectures: CIFAR10, LeNet and ImageNet. In our approach, we implement a vanilla CNN architecture and utilize broader evaluation strategies. Unlike the evaluation strategy in Afonso et al., we test: the model's performance against both *raw and filtered data*, the *generalizing power* using *leave-one-patient-out* evaluation and lastly, we comparison of the model against a *wider set of benchmarked methods*.

5.5 RPCNN architecture

RPCNN architecture is designed to leverage and complement the strengths of RPs and CNNs.It consists of two blocks (refer figure 5.1):

1. RP block

2. CNN block

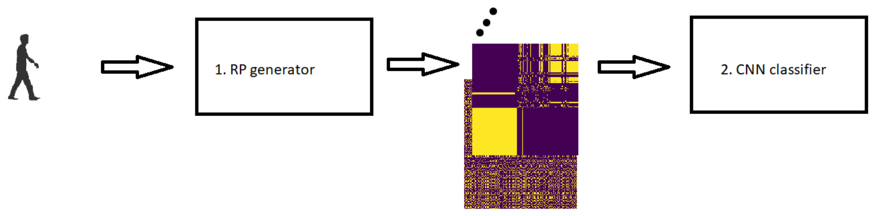


Figure 5.1: Data pipeline of our approach.

The RP block is responsible for creating RPs of the input data. Once the RPs are generated, the CNN block is responsible for training the CNN model. Since the strength of CNNs lies in image classification tasks, the authors theorize that CNN can "learn" complex non-linear patterns from the generated RPs.

5.6 Parameters and hyperparameters of the RPCNN architecture

As the RPCNN architecture consists of both RPs and CNNs, there are both "trainable" and "untrainable" parameters present in the architecture. To maintain the consistency of nomenclature, we will refer to the "trainable" parameters as *parameters* and the "untrainable" parameters as *hyperparameters*. The parameters of an RPCNN architecture are updated in the *training* process of the CNN. Whereas, the hyperparameters need to be brute-force tested, to select the best performing hyperparameters. The parameters of the RPCNN architecture are the *weights* and *biases* of the neural network. The hyperparameters of the RPCNN are: *window size of the data* for the RP, *number of neurons*, *number of layers of neurons*, *activation functions*, *regularization rate*, *learning rate*, *momentum of the learning rate* and *kernel size* for the CNN[28].

5.7 Evaluating the RPCNN architecture

The premise of the research work is to develop a PD identification technique which is not only robust to the *presence of outliers* but also includes *automatic feature engineering*. Hence the evaluation of the architecture includes: performance comparison of *raw versus filtered data*, testing the *generalizing power* of the model and *comparison against benchmarks*.

5.7.1 Raw versus filtered data

To study the effect of noisy data, the authors evaluate the performance the architecture using both noisy and filtered data. By comparing the performance metrics for both raw and filtered data, the authors aim to evaluate the effectiveness of the architecture in handling noisy data.

5.7.2 Hyperparameter tuning

As mentioned before, the architecture has hyperparameters which need to be optimized for selecting the best performing hyperparameters. In this study, the authors evaluate the performance of two hyperparameters: *window size of the data* for the RP, *number of layers* in the CNN. As this is the first research work to study the of the influence of noisy data on an RPCNN architecture, the authors evaluate two hyperparameters: *window sizes of the data* and *number of layers* in the CNN. The reason for considering *window sizes of the data* and *number of layers* in the CNN is because these two hyperparameters directly influence the time taken for generating diagnosis results. The time utilized for training the architecture is dependent on the *number of layers* in the CNN. If the number of layers is larger, then the number of parameters is bigger. For a larger number of parameters, the time taken for training the is slower, and this results in a slower diagnosis. By testing the various *window sizes of the data*, the authors evaluate the time taken for a prediction using the proposed architecture. Ideally, shorter prediction times are useful as it corresponds to a *quicker* diagnosis.

5.7.3 Generalizing power of the model

Testing the generalizing power of any ML model is an important evaluation metric. If a model can perform well on unseen data (the portion of data not used for training the model), then we say that the model has good *generalizing power*. We achieve this by performing a leave-one-subject-out evaluation similar to the research work of San-Segundo et al.[46] and Xia et al.[56]. In this method, there are two types of evaluation: *in-subject* and *cross-subject* evaluation. In *cross-subject* evaluation, data regarding one subject is used as testing, and the rest of the data is used for training the parameters of the model. In *in-subject* evaluation, data regarding the subject is used for both training and testing the model. This is shown in the figure below:

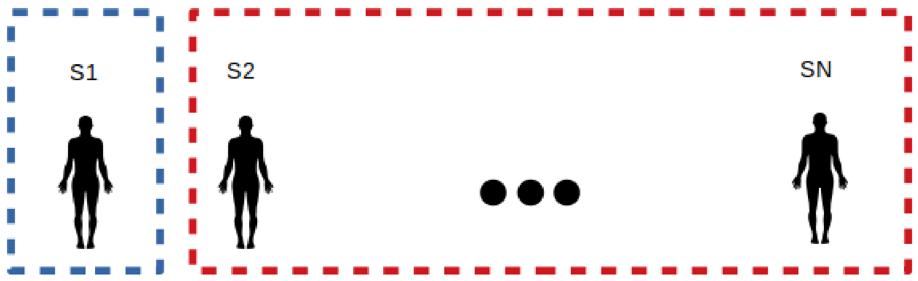


Figure 5.2: *Cross-subject* evaluation for subject "S1".

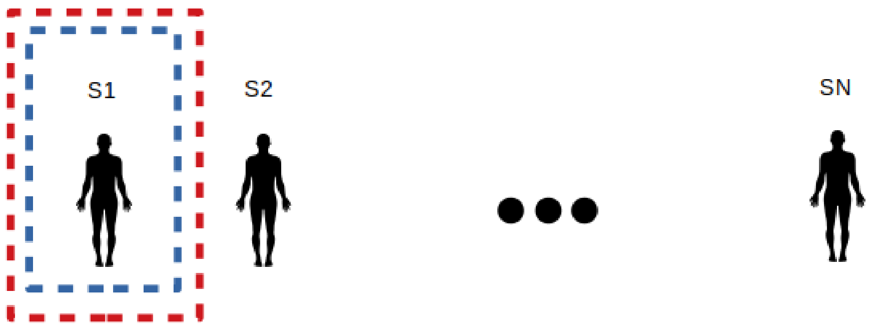


Figure 5.3: *In-subject* evaluation for subject "S1".

5.7.4 Benchmark comparison

For the performance comparison of the model, the authors have selected 5 research works as benchmark models. The authors compare the performance of the model against benchmarked methods.

How the benchmarked literature was selected

From the list of research works in table 3.2, the authors selected 5 research works as a benchmark for performance evaluation of RPCNN. The benchmark papers are represented by an asterisk in the "Study" column. The authors selected a study as a benchmark only if it satisfied both of the following criteria:

1. **Availability of original implementation code**: The implementation code for

the research work was either publicly available or the authors were able to contact the corresponding researchers for the code.

2. **Different/New algorithm for identification**: In comparison to the other research works at the time of publishing, the research work utilizes a *different/ new* identification technique .

3. **Comparable performance metrics values**: The performance metrics of the research work are close to/ better than the state-of-the-art results (at the time of publication)

The methodology of the 5 benchmarked research works is implemented and compared against the proposed RPCNN algorithm.

5.8 Drawbacks of the proposed architecture

As the proposed architecture utilizes RPs and CNNs, there are certain drawbacks emanating from the individual architectures of RPs and CNNs.

Missing values

The research work aims to reduce the dependency on handcrafted feature engineering and specific outlier treatment. Apart from outlier treatment and noise removal, another essential part of data pre-processing is *dealing with missing values* in the data. One of the significant challenges in an RP representation of data is the presence of missing values. The data in real life is imperfect and always contains noise, outliers and missing values. In chapter 2, the authors show how an RP representation of data is resilient to noise and outliers but fails in the presence of missing values. The inability of RPs to deal with missing data is reflected in proposed RPCNN architecture.

Hyperparameter tuning

The researchers aim to reduce the dependency on *complex* and *arduous* hand-crafted feature engineering techniques for PD identification. ANNs, such as CNNs, can automatically extract features, thereby negating the requirement of hand-crafted feature engineering. However, the challenge in designing any CNN is the number of hyperparameters associated with it. Because of the large number of hyperparameters and an even larger number of hyperparameter combinations, hyperparameter tuning can transform into a long-drawn process. Thus, the training of the CNNs with various hyperparameters can become an arduous process.

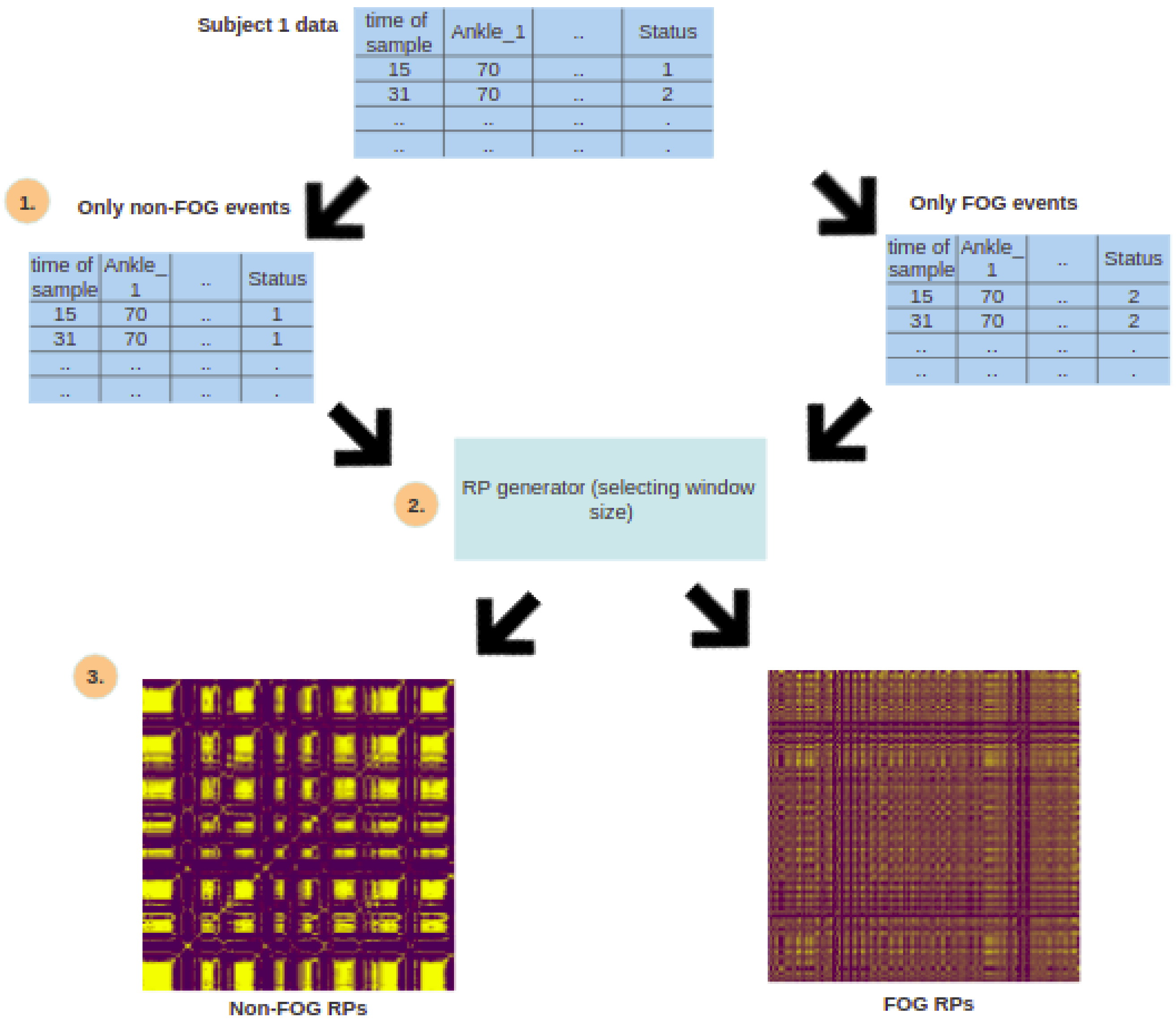
Chapter 6

Experimental Setup and Evaluation Metrics

In this chapter, the authors describe the architecture of RPCNN in greater detail, followed by an explanation of the various evaluation techniques.

6.1 Removing data not part of experiment

The data collected by Bachlin et al. contains 3 categories labelled as "FOG" (value: 2 in the dataset), "non-FOG"(value: 1 in the dataset) and "not a part of the experiment" (value: 0 in the dataset)[7]. We only considered "FOG" and "non-FOG" data, and removed the data which was not part of the experiment. Hence, we include only the "FOG" and "non-FOG" data for all the ten subjects. Visual representation of the filtering criteria is represented in figure 6.1. The filtration of the data, to include only "FOG" and "non-FOG" events,



Subject 1 data

time of sample	Ankle_1	..	Status
15	70	..	1
31	70	..	2
..	..	..	.
..	..	..	.

Only non-FOG events

time of sample	Ankle_ 1	..	Status
15	70	..	1
31	70	..	1
..	..	..	.
..	..	..	.

Only FOG events

time of sample	Ankle_ 1	..	Status
15	70	..	2
31	70	..	2
..	..	..	.
..	..	..	.

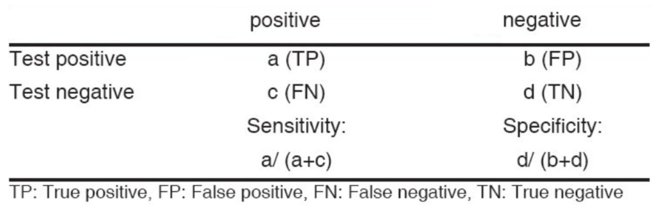
Figure 6.1: We considered only FOG and non-FOG events for our experiments.

6.2 Oversampling

As mentioned in chapter 4, the data available for "FOG" events are smaller in comparison to "non-FOG" data. Hence, to address this issue of class imbalance, the authors utilise the concept of "oversampling"[10]. In oversampling, we make multiple copies of the class with a lower number of data-points. In our case, the number of data-points for "FOG" events is much smaller than the "non-FOG" events. Hence, multiple copies of "FOG" events are made to address the issue of class imbalance.

6.3 Performance metrics

As the dataset deals with classification, the authors calculate the *sensitivity*, *specificity* and *accuracy* of the models. The calculation of *sensitivity*, *specificity* and *accuracy* is visualized in the figure 6.2



	positive	negative
Test positive	a (TP)	b (FP)
Test negative	c (FN)	d (TN)
	Sensitivity:	Specificity:
	a/ (a+c)	d/ (b+d)

TP: True positive, FP: False positive, FN: False negative, TN: True negative

Figure 6.2: Calculation of sensitivity, specificity and accuracy for our model performance.

6.4 Performance on raw versus filtered data

The premise of RPCNN is to negate the requirement of dedicated outlier treatment and feature engineering in akinesia detection. Hence, the authors test the effectiveness of RPCNN on both filtered and unfiltered data. For generating the filtered data, the authors followed the same method as Bachlin et al. The researchers in Bachlin et al., used a low pass filter to extract frequency components corresponding to (0Hz-8Hz).

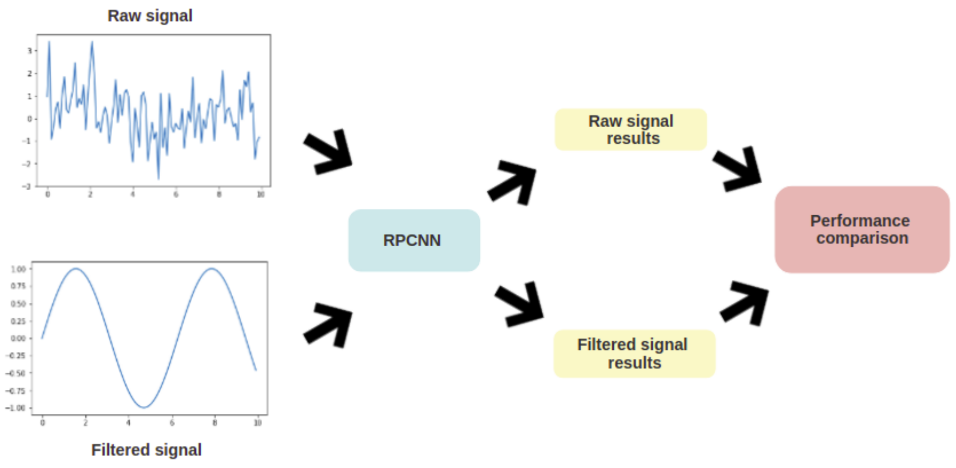


Figure 6.3: Performance comparison of raw vs filtered data on RPCNN.

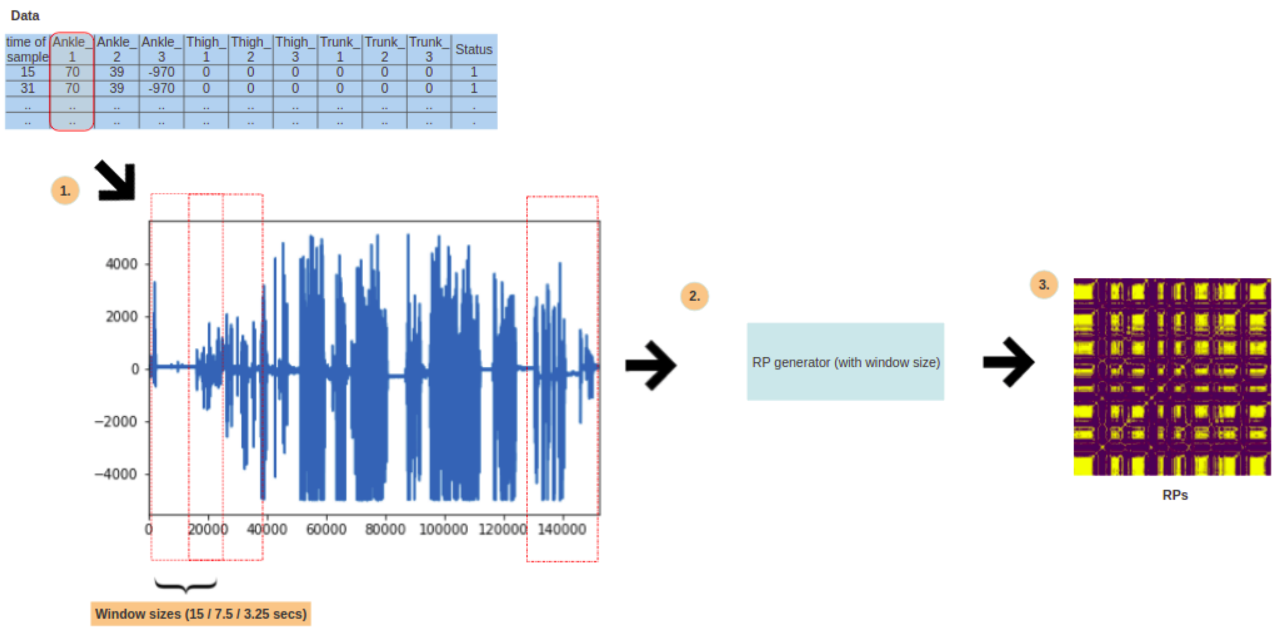
6.5 Hyperparameter tuning

As mentioned in chapter 5, the authors evaluate two different hyperparameters of the architecture: *window size of the input data* and *number of layers in the CNN*.

6.5.1 Sliding window across the data

As mentioned in chapter 2, RPs are generated for fixed time intervals. Hence, we are at liberty to select the time intervals for which the RPs are generated. The authors considered 3 different window lengths to segregate the time series data. By segregating the data into different windows, the authors aim to study the effect of window size on the architecture's performance. The 3 different window sizes considered are: 15, 7.5 and 3.25 seconds (refer figure6.4. These window sizes correspond to the average duration of FOG observed in the studies by Sawada et al. [47] and Kwon et al. [27].

Due to the 3 window sizes, we have different RP resolutions. Hence, separate CNN architecture is used for each of the RP type (15/7.5/3.25s).


Data

time of sample	Ankle_1	Ankle_2	Ankle_3	Thigh_1	Thigh_2	Thigh_3	Trunk_1	Trunk_2	Trunk_3	Status
15	70	39	-970	0	0	0	0	0	0	1
31	70	39	-970	0	0	0	0	0	0	1
..	..	..	..	..	..	..	..	..	..	.
..	..	..	..	..	..	..	..	..	..	.

Figure 6.4: We calculated RPs for various sliding windows.

6.5.2 "Deep" vs "Shallow" RPCNN

As mentioned in chapter 5, the authors aim to evaluate the complexity of the architecture. The complexity of RPCNN is studied by evaluating the performance of "shallow" and "deep" architectures. As mentioned before, this is the first research work to study the influence of noisy data on an RPCNN architecture. Hence, the authors implement and evaluate a relatively *simplistic* version of the vanilla CNN model. The model has between 1-to-4 convolutional layers, followed by ReLU activation and ending in 2 fully connected layers(refer figure 6.5).

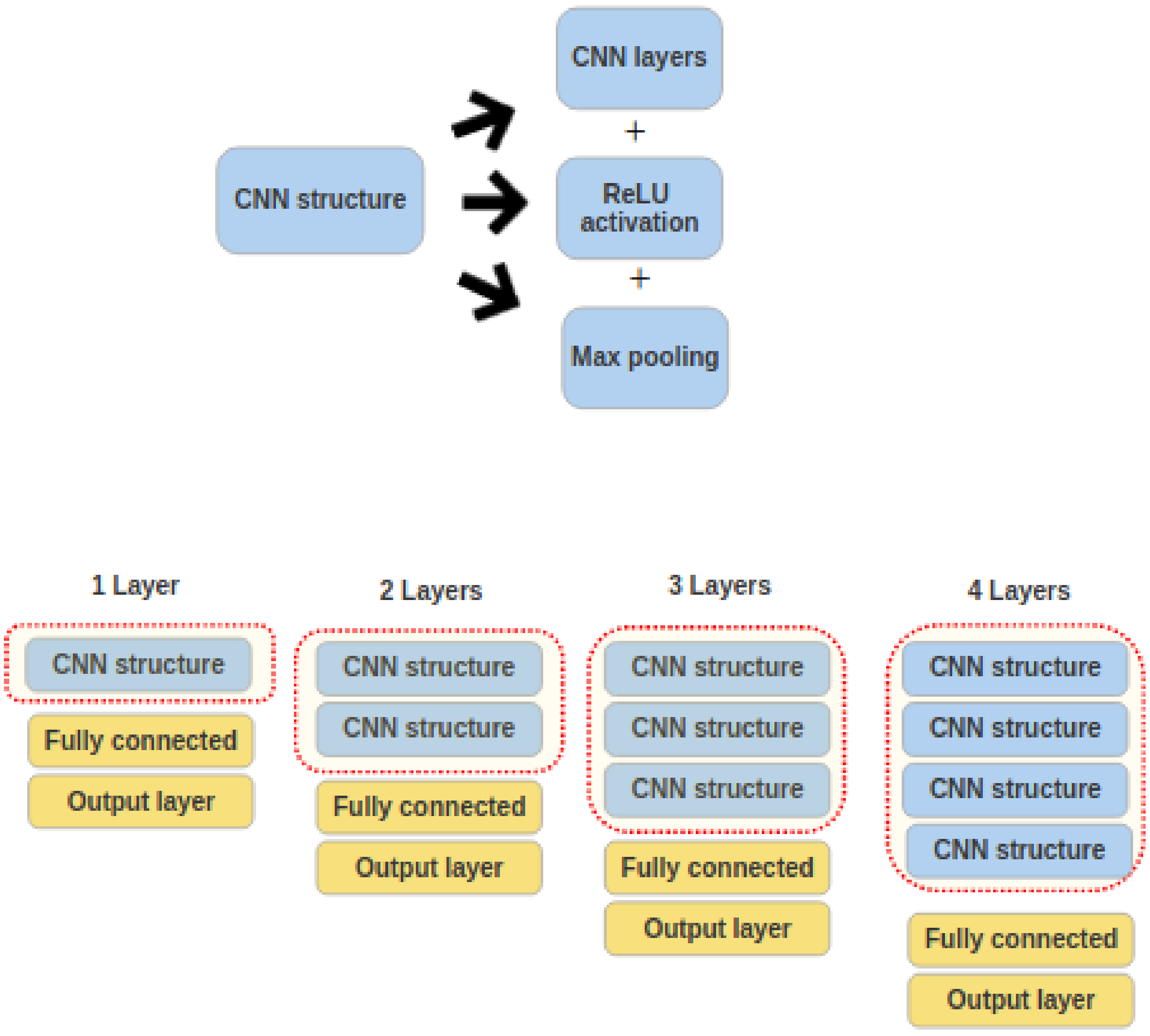


Figure 6.5: Varying depth of CNN for performance comparison.

In the "1 layer" CNN structure, we use a single stack of CNN kernels (with ReLU activation) and max-pooling kernels. In a "2 Layer" structure will have two stacks of CNN and max-pooling kernels followed by the fully connected layer and the output layer. The example of an RPCNN architecture with 1 CNN layer is represented in table 6.1.

Table 6.1: Example of the CNN classifier block with 1 stack of CNN layers

Section	**Type**	**Kernel size/ Neurons (for fully connected)**	**Stride**
1	Conv with ReLU	5 x 5	1
	Max-Pooling	2 x 2	1
2	Fully connected	50	NA
3	Output Layer	1	NA

A "deeper" RPCNN architecture corresponds to a large number of CNN layers. The example of an RPCNN architecture with 2 CNN layers is represented in table 6.2.

Table 6.2: Example of the CNN classifier block with 2 stacks of CNN layers

Section	**Type**	**Kernel size/ Neurons (for fully connected)**	**Stride**
1	Conv with ReLU	5 x 5	1
	Max-Pooling	2 x 2	1
2	Conv with ReLU	5 x 5	1
	Max-Pooling	2 x 2	1
3	Fully connected	50	NA
4	Output Layer	1	NA

6.6 In-subject and cross-subject evaluation

Apart from the accuracy metrics and parameter tuning, the authors also followed an evaluation strategy inspired by [42] and [33]. To study the *generalising power* of the RPCNN architecture, we define two types of modelling evaluations(figure

6.6):

1. **In-subject evaluation**: Here, the data of a single person is used as both testing and training data. For example, for subject 1, only the data corresponding to subject 1 is used as both training and testing data.

2. **Cross-subject evaluation**: Here, the data related to one person is used as testing data, and the rest of the data is the training data. Hence, if we are calculating the cross-patient performance for subject 1, the data of subject 1 will be used as testing data and data for subjects 2 through 10 (subject 2, subject 3...... and subject 10) will be used as training data.

This form of in-subject and cross-subject evaluation technique evaluates the generalising power of the model. In other words, if the model is identifying patterns on the *unseen data* as well as the *seen* data. An ideal model performs well on both in-subject and cross-subject evaluation.

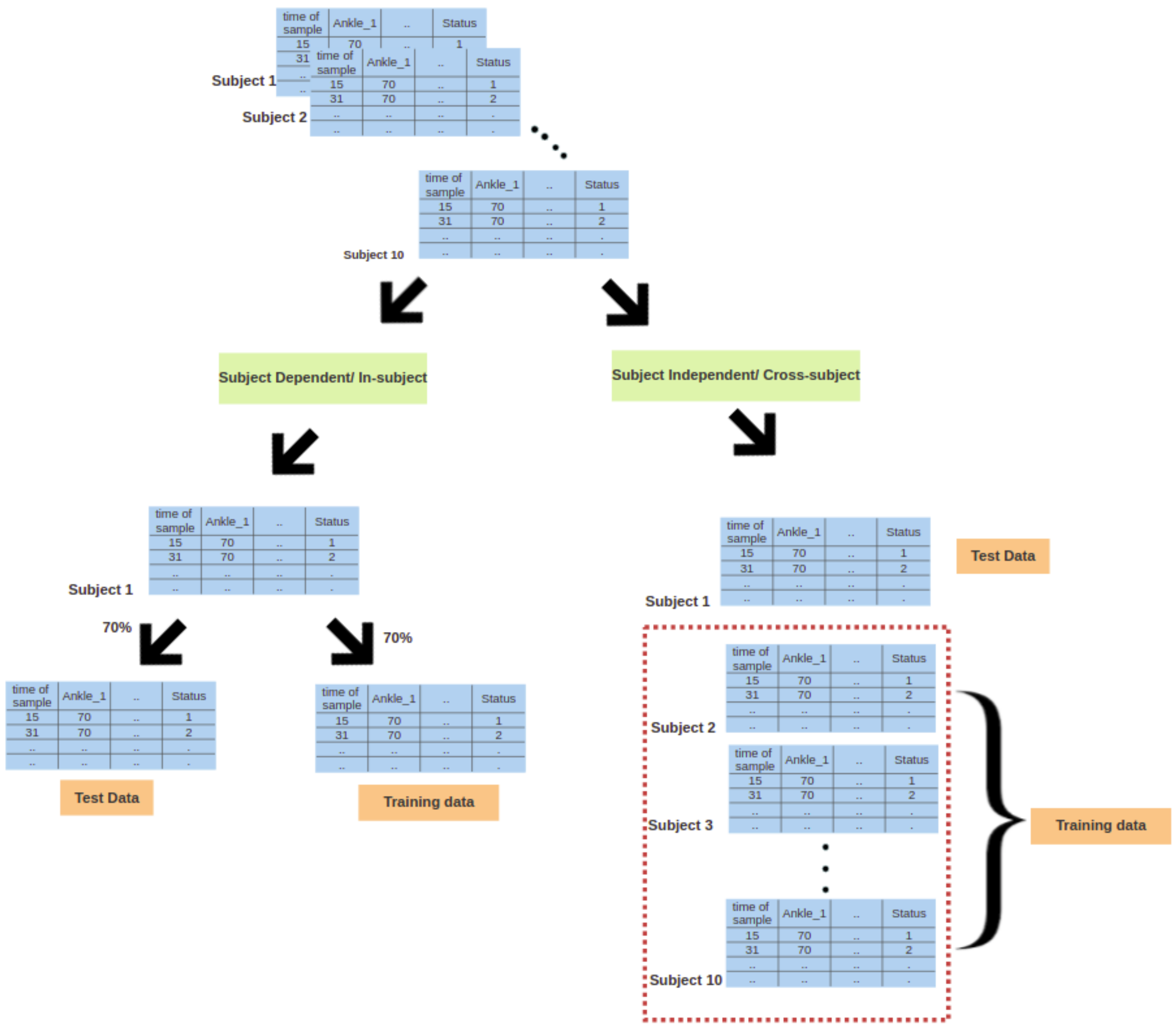


Figure 6.6: Subject dependent and subject independent approaches to evaluate performance of the model for subject 1 as an example.

6.7 Benchmark comparison

As mentioned in chapter 5, the authors select 5 research works as *benchmarks*, for comparison against the proposed approach. The reasons for selecting the 5 research work as benchmarks have been addressed in chapter 5. The implementation code of Bachlin et al. was publicly available[7] but needed a few changes dues to the presence of old libraries. The implementation code for Ahlrichs et al.[3], Kim et al.[24], Alsheikh et al.[6] and Ravi et al.[40] was obtained by contacting the respective researchers.

Bachlin et al. uses the research of Moore et al.[34] to calculate the entropy in the two frequency bands (0.5Hz-3Hz) and (3H-8Hz). After empirically calculating the *threshold* for the "FOG" and "non-FOG" indexes, they apply the thresholding algorithm to discern "FOG" events.

In Ahlrichs et al., 8 statistical and frequency domain features are extracted in conjunction with a "freezing index"[3]. The "freezing index" is encoded via a thresholding algorithm which is similar to the approach of Bachlin et al.[7]. Ahlrichs et al. use a low pass filter to include data corresponding to (0Hz-8Hz) band. Along with the features specified by Bachlin et al., Kim et al. create more statistical and frequency domain features in the (0.5Hz-3Hz) and (3H-8Hz)[24]. Alsheikh et al.[6] and Ravi et al.[40] use a spectrogram representation of the data as inputs to the classifier. Alsheikh et al. use a variation of the deep belief networks (DBNs) and restricted boltzmann machines (RBM) for activity recognition.

Ravi et al. use a CNN to classify the spectrogram representation of the input.

As we can see, the 5 benchmarked methods include various classification models in conjunction with different feature extraction techniques.

Table 6.3: List of benchmarked methods.

Study	**Year**	**Reference**	**Sample**	**Sensors**	**Feature engineering/Outlier treatment (OT)?**	**algorithm**	**Results**
Bachlin et al.	2010	[7]	10 PD	accelerometers	OT + frequency features	Energy Thresholding	mean sensitivity and specificity of 73.1% and 81.6%
Ahlrichs et al.	2016	[3]	20 PD	accelerometers + gyroscopes	OT + frequency features	SVM	classification accuracy of 95.4%
Kim et al.	2015	[24]	15 PD	accelerometers	OT + statistical features + frequency features	RF	highest classification sensitivity and specifiaccelerometerscity of 86% and 91.7%
Alsheikh et al.	2016	[6]	10 PD	accelerometers	frequency features	DBN	highest classification accuracy of 97.85%
Ravi et al.	2016	[40]	10 PD	accelerometers	frequency features	CNN	sensitivity and -specificity of 91.5% and 97.7%

Chapter 7

Results

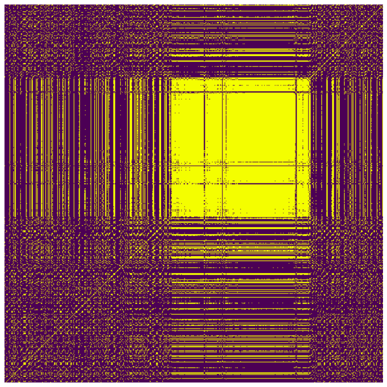
In this chapter, the authors mention and discuss the results obtained. We first look at the resulting RPs for a small sample of data. Then we proceed to compare the performance of RPCNN against the benchmarked methods. Lastly, we look at the effect of the number of layers and window size, on the classification performance.

7.1 Recurrence Plots of the data

RPs of a sample of the data is shown below. To illustrate the visual representation of temporal signals, we view the RPs of subject 1. As mentioned in chapter 5, RPs for both raw and filtered data are calculated.

7.1.1 Recurrence Plots of the raw data

A subset of RPs for the raw signals from subject 1 data are present in figures 7.1 through 7.3. We can see that RPs for non-fog events have distinct patterns as tasks like walking, turning are periodic. The RPs for FOG events do not have any distinct patterns.


(a) Non-fog RP

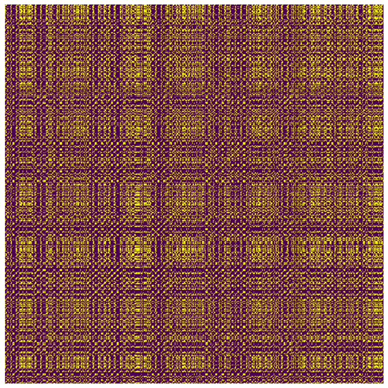
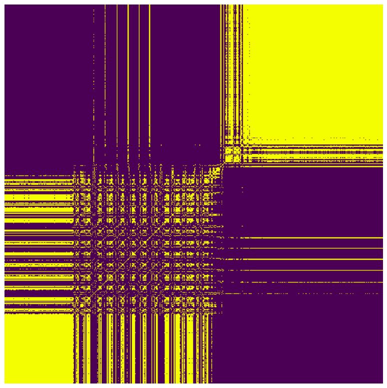

(b) Fog RP

Figure 7.1: Window size 3.25s


(a) Non-fog RP

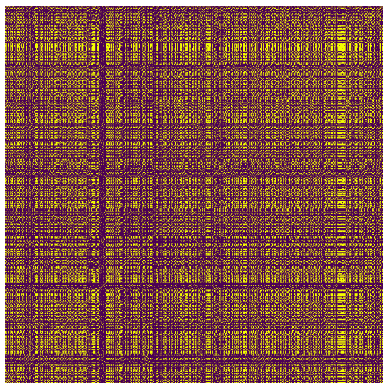

(b) Fog RP

Figure 7.2: Window size 7.5s

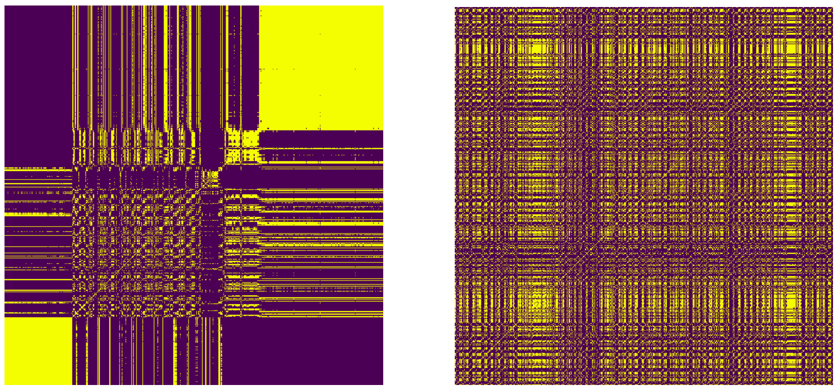

(a) Non-fog RP (b) Fog RP

Figure 7.3: Window size 15s

7.1.2 Recurrence Plots of the filtered data

A subset of RPs for the filtered signals from subject 1 data are present in figures 7.4 through 7.6. We can see that RPs for non-fog events have distinct patterns as tasks like walking, turning are periodic. The RPs for FOG events do not have any distinct patterns.

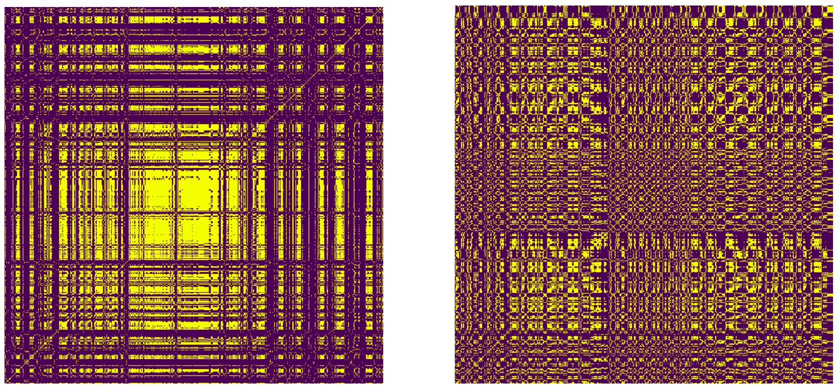

(a) Non-fog RP (b) Fog RP

Figure 7.4: Window size 3.25s, subset of patient 1 data

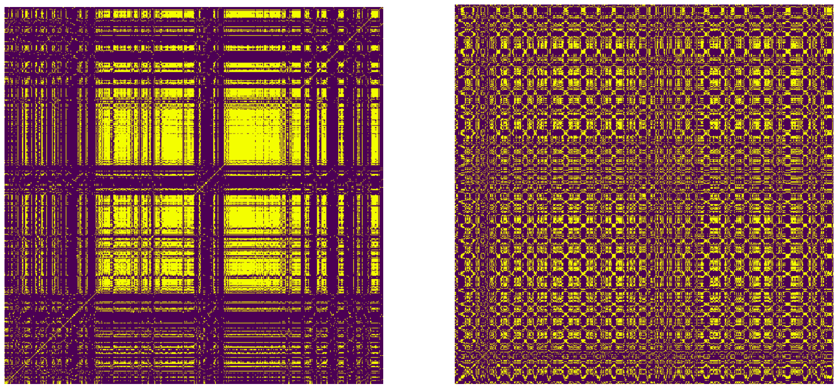

(a) Non-fog RP (b) Fog RP

Figure 7.5: Window size 7.5s, subset of patient 1 data

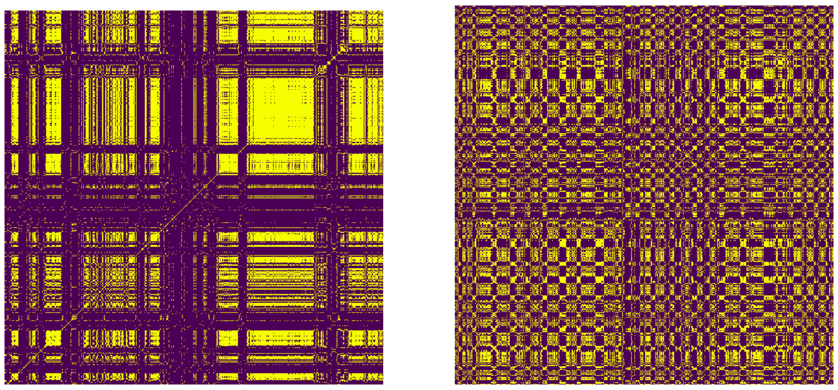

(a) Non-fog RP (b) Fog RP

Figure 7.6: Window size 15s, subset of patient 1 data

7.1.3 Observations

We can visually observe that the RPs for FOG and non-FOG events are different. These results correlate with the observations of Afonso *et al.*, as the researchers Afonso *et al.* also observed a visual difference between "resting state" and "tremor state" RPs[37]. The difference in "FOG" RPs and "non-FOG" RPs is observable irrespective of the signal being raw or filtered.

7.2 Results against benchmarks

The authors compared the performance of the proposed RPCNN model against the benchmark methods mentioned in chapter 3. Here too, the authors supplied both raw and filtered data separately to the RPCNN. If raw data is supplied to the RPCNN, it is represented by raw-RPCNN. If filtered data is supplied to the RPCNN, the resulting architecture is represented by filtered-RPCNN. By supplying both raw and filtered data separately to the RPCNN structure, the authors study the effect of outliers on RPCNN structure. As mentioned in chapter 3, the benchmarked data was selected such that the implementation code is either publicly available or the authors obtained it by contacting the respective researchers. Hence, the results of the benchmarked data are obtained by implementation on the same machine.

Table 7.1: Comparison of RPCNN against benchmarks.

Method Used	**Year**	**Feature engineering/ Outlier treatment (OT)?**	**Sensitivity**	**Specificity**
Bachlin et al.	2010	OT+ frequency domain features	88.12±7.5%	89.22±5.5%
Pham et al.	2017	OT+ frequency domain features	76.7±5.7%	84.9±7.5%
Ravi et al.	2016	frequency domain features	81.6±2.9%	82.9±3.2%
San-Segundo et al.	2019	OT+ frequency domain features	85.8±6.5%	79.6±4.3%
Alsheikh et al.	2016	frequency domain features	90.79±11.5%	87.5±8.5%
filtered- RPCNN	2019	-	81.98±4.2%	87.67±3.6%
raw -RPCNN	2019	-	69.72±8.4%	74.87±6.2%

7.2.1 RPCNN is affected by noisy data

The difference of 12.26% in sensitivity and 12.80% in the specificity of the RPCNN model suggests that it is still sensitive to the presence of noise and outliers. Even though the performance metrics of raw-RPCNN are lower than filtered-RPCNN, they are still comparable to other benchmark methods. The filtered-RPCNN performs better than raw-RPCNN; this is because filtered-RPCNN utilizes filtered data; this suggests that further research needs to be done to improve the performance using raw data.

7.3 Performance on raw versus filtered data

In this section, we look at the differences in performance metrics for raw versus filtered data. The authors first discuss the best performance architecture for raw and filtered data. The authors also discuss the effect of the parameters (number of layers and window size, in our case) on classifying raw versus filtered data.

7.3.1 Best performance on raw data

For raw data, the model performs better in in-subject evaluations than cross-subject evaluations.(refer table 7.2).

Table 7.2: Best combination of parameters for in-subject and cross-subject evaluations using raw data.

Evaluation	**Combination (window size + layers)**	**Sensitivity**	**Specificity**
In-subject	7.5s+1 layer	72.34%	81.69%
Cross-subject	7.5s+3 layers	52.35%	72.19%

7.3.2 Best performance on filtered data

For filtered data too, the model performs better in in-subject evaluations than cross-subject evaluations.(refer table 7.3).

Table 7.3: Best combination of parameters for in-subject and cross-subject evaluations using filtered data.

Evaluation	**Combination (window size + layers)**	**Sensitivity**	**Specificity**
In-subject	3.5s+1 layer	90.87%	85.43%
Cross-subject	3.5s+2 layers	72.69%	76.61%

The performance metrics for filtered data are higher than the raw data. In fact, the sensitivity value jumps from 52.35% to 72.69% if we use filtered data in cross-subject evaluation. This suggests that the model is relatively weak if raw data is directly used. The performance metric values are higher for in-subject than cross-subject; this suggests that the model is having difficulty in adapting unseen and unknown data.

7.3.3 Effect of layers on raw data classification

Table 7.4: Effect of number of layers on performance metric values for filtered data.

	Se %		Sp %		Acc %	
No. of layers	avg	std	avg	std	avg	std
1	21.45	21.73	**78.25**	31.12	59.68	27.23
2	35.66	22.67	74.97	30.43	62.62	18.42
3	**49.53**	29.12	62.19	30.18	**65.56**	21.80
4	37.23	28.27	62.41	25.21	57.56	23.87

7.3.4 Effect of layers on filtered data classification

Table 7.5: Effect of number of layers on performance metric values for raw data.

	Se %		Sp %		Acc %	
No. of layers	avg	std	avg	std	avg	std
1	**72.21**	23.28	**85.69**	35.31	**82.73**	28.33
2	65.49	29.18	73.34	21.15	75.88	31.80
3	62.95	32.13	68.30	32.27	71.94	29.37
4	59.66	28.78	68.69	25.47	69.14	28.12

For raw data, the architecture requires a higher number of layers to develop complex non-linear boundaries. In the filtered data, the shallower model performs best.

7.3.5 Effect of window size on raw data classification

Table 7.6: Effect of window size on performance metric values for raw data.

	Se %		Sp %		Acc %	
Window size	avg	std	avg	std	avg	std
15s	**49.11**	26.89	75.10	7.17	70.25	8.23
7.5s	34.41	18.36	**85.49**	13.56	**75.26**	12.80
3.25s	45.56	27.45	76.78	9.80	69.38	9.59

7.3.6 Effect of window size on filtered data classification

The results for in-subject evaluations are present in table 7.7 and table 7.5.

Table 7.7: Effect of window size on performance metric values for filtered data.

	Se %		Sp %		Acc %	
Window size	avg	std	avg	std	avg	std
15s	71.19	23.77	78.47	15.38	87.30	10.65
7.5s	62.28	17.50	80.28	19.83	81.76	14.25
3.25s	**75.59**	22.68	**86.47**	16.53	**89.59**	9.19

For raw data, the classifier performs best on 7.5 seconds of time-window. For filtered data, the best performing time-window is 3.25 seconds. In comparison to raw data classifier, filtered data classifier is preferred because the 7.5-second interval defeats the purpose of designing quick and efficient FOG detection systems. Hence, further research is required to investigate how window size affects FOG detection time.

7.3.7 Observations

1. RPCNN does not perform well on unseen data

In the case of both raw and filtered data, the RPCNN architecture has higher performance metric scores for in-subject evaluation than cross-subject evaluation. This suggests that the model performs well for seen data but fails to perform the same for unseen data.

2. RPCNN requires fewer parameters for filtered data than raw data

When using filtered data, the highest performance metric scores were observed for a single layer of CNN. However, for raw data, a higher number of layers registered higher performance metric scores. This suggests that the model requires fewer parameters for filtered data but a larger number of parameters for raw data. A larger number of parameters requires more time to train the model and proves to be a drawback as the PD detection algorithm becomes slower.

3. RPCNN requires smaller window size for filtered data than raw data

For the filtered data, the RPCNN required 3.25 seconds of window size to score the highest performance metric values. For raw data, the required window size is larger. The specificity and accuracy are highest for 7.5s; the sensitivity is highest for 15s. This suggests that for noisy data, larger input is required by the RPCNN to make a correct prediction. Larger window sizes are a major hindrance in designing quick and efficient PD identification systems because of the latency in prediction.

Chapter 8

Critical reflection and conclusion

The research work aims to negate the dependency on handcrafted feature engineering and dedicated outlier removal processes in PD identification. To achieve this goal, the authors theorize and evaluate RPCNN architecture. The performance metrics of the architecture suggests that further work is needed to improve the architecture. As part of the critical reflection on the research work, the authors observe the following:

1. **"Strengthening" the simple architecture**

As this is the first research work to explore the performance of RPCNN against noisy data, the authors could not perform extensive study on the various hyperparameters of the architecture. From the results of the *raw data evaluation* in chapter 7, we see that higher number of layers give a better performance. This suggests that model may be *too simple* in terms of the CNN architecture, for noisy data. Hence, further work can be done to include and study the various hyperparameters such as: *learning rate*, *momentum*, *regularization rate*, *activation functions*, *kernel size*, to name a few.

2. **Evaluating standard architectures**

Along the similar lines of increasing the complexity, the CNN block of the model can be "beefed" up by implementing the standard CNN image classifiers. In Afonso et al., the researchers train standard CNN image classifiers: ImageNet, CIFAR-10 and VGG[37]. A similar approach can be used to study the effect of standard CNN classifiers on RP representation of data.

3. **Empirical search of ε value**

As mentioned before, this is the first research work to explore the performance of RPCNN against noisy data. Hence, the authors followed the findings of Marwan et al. to calculate the value of ε. Marwan et al. suggest using 10% of the mean as the threshold value (ε) in calculating RPs[30]. Further work can be done to study the effect of various ε values on the performance metrics.

4. **Missing data**

The premise of the research work is to negate the requirement of handcrafted feature engineering and dedicated outlier removal. But, another important step in *data pre-processing* is the treatment of missing values. One of the major drawbacks of the proposed architecture is the inability to address missing values in the data properly. In practical applications, the data is not always *clean* and often has missing values. As the RPs cannot treat missing data, the RPCNN architecture fails at missing data.

RPs, in conjunction with CNNs, has opened a new area of research where automatic feature extraction is robust against the presence of outliers. By improving the current RPCNN architecture, we move a step ahead in designing fully automated systems. The implementation of such an automated architecture will not be restricted to PD detection but can be applied to many other domains.

Bibliography

[1] Luis CS Afonso et al. “A recurrence plot-based approach for Parkinsonfffdfffdfffds disease identification”. In: *Future Generation Computer Systems* 94 (2019), pp. 282–292.

[2] Megha Agrawal and Abhijit Biswas. “Molecular diagnostics of neurodegenerative disorders”. In: *Frontiers in molecular biosciences* 2 (2015), p. 54.

[3] Claas Ahlrichs et al. “Detecting freezing of gait with a tri-axial accelerometer in Parkinson’s disease patients”. In: *Medical & biological engineering & computing* 54.1 (2016), pp. 223–233.

[4] Mark V Albert et al. “Using mobile phones for activity recognition in Parkinson’s patients”. In: *Frontiers in neurology* 3 (2012), p. 158.

[5] Noura AlNuaimi et al. “Streaming Feature Selection Algorithms for Big Data: A Survey”. In: *Applied Computing and Informatics* (2019).

[6] Mohammad Abu Alsheikh et al. “Deep activity recognition models with triaxial accelerometers”. In: *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[7] Marc Bächlin et al. “Wearable assistant for Parkinson’s disease patients with the freezing of gait symptom.” In: *IEEE Trans. Information Technology in Biomedicine* 14.2 (2010), pp. 436–446.

[8] Liviu Badea et al. “Exploring the reproducibility of functional connectivity alterations in Parkinson’s disease”. In: *PloS one* 12.11 (2017), e0188196.

[9] Nooshin Bigdeli et al. “A comparative study of optimal hybrid methods for wind power prediction in wind farm of Alberta, Canada”. In: *Renewable and sustainable energy reviews* 27 (2013), pp. 20–29.

[10] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[11] Heiko Braak et al. “Staging of brain pathology related to sporadic Parkinson’s disease”. In: *Neurobiology of aging* 24.2 (2003), pp. 197–211.

[12] Julia Camps et al. “Deep learning for freezing of gait detection in Parkinson’s disease patients in their homes using a waist-worn inertial measurement unit”. In: *Knowledge-Based Systems* 139 (2018), pp. 119–131.

[13] J Cancela et al. “A comprehensive motor symptom monitoring and management system: the bradykinesia case”. In: *2010 Annual International Con-*

ference of the IEEE Engineering in Medicine and Biology. IEEE. 2010, pp. 1008–1011.

[14] Diane J Cook, Maureen Schmitter-Edgecombe, and Prafulla Dawadi. "Analyzing activity behavior and movement in a naturalistic environment using smart home techniques". In: *IEEE journal of biomedical and health informatics* 19.6 (2015), pp. 1882–1892.

[15] JP Eckmann, S Oliffson Kamphorst, D Ruelle, et al. "Recurrence plots of dynamical systems". In: *World Scientific Series on Nonlinear Science Series A* 16 (1995), pp. 441–446.

[16] Alessandro Giuliani et al. "Nonlinear signal analysis methods in the elucidation of protein sequence- structure relationships". In: *Chemical Reviews* 102.5 (2002), pp. 1471–1492.

[17] JI Hoff et al. "Accelerometric assessment of levodopa-induced dyskinesias in Parkinson's disease". In: *Movement disorders: official journal of the Movement Disorder Society* 16.1 (2001), pp. 58–61.

[18] David H Hubel and Torsten N Wiesel. "Receptive fields of single neurones in the cat's striate cortex". In: *The Journal of physiology* 148.3 (1959), pp. 574–591.

[19] Diego Iacono et al. "Parkinson disease and incidental Lewy body disease: just a question of time?" In: *Neurology* 85.19 (2015), pp. 1670–1679.

[20] Y Nancy Jane, H Khanna Nehemiah, and Kannan Arputharaj. “A Q-backpropagated time delay neural network for diagnosing severity of gait disturbances in Parkinson’s disease”. In: *Journal of biomedical informatics* 60 (2016), pp. 169–176.

[21] Joseph Jankovic. “Parkinson’s disease: clinical features and diagnosis”. In: *Journal of neurology, neurosurgery & psychiatry* 79.4 (2008), pp. 368–376.

[22] Samuel Kaski et al. “WEBSOM–self-organizing maps of document collections”. In: *Neurocomputing* 21.1-3 (1998), pp. 101–117.

[23] Noël LW Keijsers, Martin WIM Horstink, and Stan CAM Gielen. “Automatic assessment of levodopa-induced dyskinesias in daily life by neural networks”. In: *Movement disorders: official journal of the Movement Disorder Society* 18.1 (2003), pp. 70–80.

[24] Hanbyul Kim et al. “Unconstrained detection of freezing of Gait in Parkinson’s disease patients using smartphone”. In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2015, pp. 3751–3754.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[26] Jürgen Kurths et al. “Testing for nonlinearity in radiocarbon data”. In: *Nonlinear Processes in Geophysics* 1.1 (1994), pp. 72–76.

[27] Yuri Kwon et al. “A practical method for the detection of freezing of gait in patients with Parkinson’s disease”. In: *Clinical interventions in aging* 9 (2014), p. 1709.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), p. 436.

[29] Max A Little et al. “Suitability of dysphonia measurements for telemonitoring of Parkinson’s disease”. In: *IEEE transactions on bio-medical engineering* 56.4 (2009), p. 1015.

[30] Norbert Marwan et al. “Recurrence plots for the analysis of complex systems”. In: *Physics reports* 438.5-6 (2007), pp. 237–329.

[31] Norbert Marwan et al. “Recurrence plots for the analysis of complex systems”. In: *Physics reports* 438.5-6 (2007), pp. 237–329.

[32] Norbert Marwan et al. “Recurrence-plot-based measures of complexity and their application to heart-rate-variability data”. In: *Physical review E* 66.2 (2002), p. 026702.

[33] Sinziana Mazilu et al. “Online detection of freezing of gait with smartphones and machine learning techniques”. In: *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*. IEEE. 2012, pp. 123–130.

[34] Steven T Moore, Hamish G MacDougall, and William G Ondo. "Ambulatory monitoring of freezing of gait in Parkinson's disease". In: *Journal of neuroscience methods* 167.2 (2008), pp. 340–348.

[35] Fernando L Pagan. "Improving outcomes through early diagnosis of Parkinson's disease." In: *The American journal of managed care* 18.7 Suppl (2012), S176–82.

[36] Shyamal Patel et al. "Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors". In: *IEEE transactions on information technology in biomedicine* 13.6 (2009), pp. 864–873.

[37] Clayton R Pereira et al. "Handwritten dynamics assessment through convolutional neural networks: An application to Parkinson's disease identification". In: *Artificial intelligence in medicine* 87 (2018), pp. 67–77.

[38] Thuy T Pham et al. "An anomaly detection technique in wearable wireless monitoring systems for studies of gait freezing in Parkinson's disease". In: *2017 International Conference on Information Networking (ICOIN)*. IEEE. 2017, pp. 41–45.

[39] Marios Politis et al. "Parkinson's disease symptoms: the patient's perspective". In: *Movement Disorders* 25.11 (2010), pp. 1646–1651.

[40] Daniele Ravi et al. "Deep learning for human activity recognition: A resource efficient implementation on low-power devices". In: *2016 IEEE*

13th International Conference on Wearable and Implantable Body Sensor Networks (BSN). IEEE. 2016, pp. 71–76.

[41] Saba Rezvanian and Thurmon Lockhart. “Towards real-time detection of freezing of gait using wavelet transform on wireless accelerometer data”. In: *Sensors* 16.4 (2016), p. 475.

[42] Daniel Rodríguez-Martín et al. “Home detection of freezing of gait using support vector machines through a single waist-worn triaxial accelerometer”. In: *PloS one* 12.2 (2017), e0171764.

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

[44] Serge H Roy et al. “Resolving signal complexities for ambulatory monitoring of motor function in Parkinson’s disease”. In: *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2011, pp. 4836–4839.

[45] Ali Saad et al. “A preliminary study of the causality of freezing of gait for parkinson’s disease patients: Bayesian belief network approach”. In: *International Journal of Computer Science Issues* 10.3 (2013), pp. 88–95.

[46] Rubén San-Segundo et al. “Increasing Robustness in the Detection of Freezing of Gait in Parkinson’s Disease”. In: *Electronics* 8.2 (2019), p. 119.

[47] Makoto Sawada et al. “Clinical features of freezing of gait in Parkinson’s disease patients”. In: *Brain and behavior* 9.4 (2019), e01244.

[48] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[49] Evanthia E Tripoliti et al. “Automatic detection of freezing of gait events in patients with Parkinson’s disease”. In: *Computer methods and programs in biomedicine* 110.1 (2013), pp. 12–26.

[50] Markos G Tsipouras et al. “Automated Levodopa-induced dyskinesia assessment”. In: *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE. 2010, pp. 2411–2414.

[51] Paul J Tuite, Silvia Mangia, and Shalom Michaeli. “Magnetic resonance imaging (MRI) in Parkinson’s disease”. In: *Journal of Alzheimer’s disease & Parkinsonism* (2013), p. 001.

[52] Richard L Villars, Carl W Olofson, and Matthew Eastwood. “Big data: What it is and why you should care”. In: *White Paper, IDC* 14 (2011), pp. 1–14.

[53] Ferdous Wahid et al. “Classification of Parkinson’s disease gait using spatial-temporal gait features”. In: *IEEE journal of biomedical and health informatics* 19.6 (2015), pp. 1794–1802.

[54] Ian H Witten et al. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[55] Yunfeng Wu et al. “Measuring signal fluctuations in gait rhythm time series of patients with Parkinson’s disease using entropy parameters”. In: *Biomedical Signal Processing and Control* 31 (2017), pp. 265–271.

[56] Yi Xia et al. “Evaluation of deep convolutional neural networks for detection of freezing of gait in Parkinson’s disease patients”. In: *Biomedical Signal Processing and Control* 46 (2018), pp. 221–230.

[57] Joseph P Zbilut et al. “Use of recurrence plots in the analysis of heart beat intervals”. In: *[1990] Proceedings Computers in Cardiology*. IEEE. 1990, pp. 263–266.

[58] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[59] Daphne GM Zwartjes et al. “Ambulatory monitoring of activities and motor symptoms in Parkinson’s disease”. In: *IEEE transactions on biomedical engineering* 57.11 (2010), pp. 2778–2786.